The Digital Migrant

a guide to computers for the curious and confused

DANU POYNER

The Digital Migrant

a guide to computers for the curious and confused

by Danu Poyner

Copyright © 2009 Danu Poyner

Cover design by Wild Witch Graphics
Book design by Danu Poyner

All rights reserved.

No part of this book may be reproduced in any form or by any electronic or mechanical means including information storage and retrieval systems, without permission in writing from the author. The only exception is by a reviewer, who may quote short excerpts in a review.

Published by Danu Poyner
Contact details at www.danupoyner.com

Visit www.thedigitalmigrant.com for more information and updates

Printed and distributed by lulu.com
ISBN 978-1-4092-8861-9

First Printing: July 2009

For my Granddad, whose generosity and enthusiasm sparked a lifelong passion.

Contents

Introduction 7

PART I The Nitty Gritty

Chapter 1

10 Kinds of People 14

in which we discover the origins of computers and discuss their basic building blocks: bits, bytes and binary

Chapter 2

Ins and Outs 36

in which we learn how a computer thinks and in which memory, processing power and operating systems are explained

Chapter 3

The Cloud 68

in which we explore the nature of the internet and learn how search engines work

PART II: The Real World

Chapter 4

Turf Wars 106

in which we explore the real differences between Windows, Mac and Linux operating systems and why compatibility problems exist

Chapter 5

Mostly Harmless 142

in which viruses and all manner of other computer security problems are explained

Chapter 6

Six Degrees 179

in which we discover how computers are changing our world and helping society reconnect

Afterword 219

Acknowledgments 224

Appendix 229

Introduction

I was lucky. I grew up with computers. I have my Granddad to thank for this. He cannot resist fiddling with things, pulling them apart to see how they work and putting them together again in different ways. This did not always please my mother who as a little girl lost many cherished toys to Granddad's hands-on curiosity, but to me it was fascinating, and as the only child of an only child I was the recipient of many interesting gadgets once Granddad became bored with them.

In the late eighties Granddad bought a personal computer. It was an Amstrad, with two giant 5.25 inch floppy drives and a black and white monitor. Among other things it ran a game called Digger in which you controlled a little bulldozer who ran around the screen collecting gems and avoiding nobbins, who followed the tracks you had dug, and so-called hobbins, who could chew through walls and were therefore much scarier.

It also had a game called Rogue, in which you moved a character through different levels of a dungeon collecting items and fighting creatures such as hobgoblins, represented on-screen in the terrifying form of a capital H. I always enjoyed trips to my grandparents' house so I could lock myself away for hours and play these games, occasionally calling out for mum to help when I encountered a particularly dangerous looking consonant. I was 5 years old.

A couple of years later Granddad bought a new computer. This one had a colour screen and an incredible 40 megabyte hard drive. I wasn't sure what to be more excited about - the new computer or the fact Granddad was giving me his old one! It soon turned out I was equally excited by both. And so the process continued, every couple of years Granddad would get a new computer with all the latest additions and I would inherit the old one. This is how I grew up. In the late eighties and early nineties this was fairly rare.

Fast forward to age 15. I'm in high school, waiting 20 minutes for my English lesson to start because my teacher can't work out how to print an email. I remember feeling frustrated by what seemed to me to be an obvious problem with a simple solution.

Computers and the technology associated with them were obviously becoming a big part of the future. School was supposed to be preparing us for the future but the people teaching us weren't prepared for the future themselves.

The teachers needed an education. I decided to run a series of after-school classes for teachers who wanted to better understand computers. My approach was to start right at the beginning and explain the binary number system and how all computers operated using only a glorified series of ones and zeroes. Several of my teachers, now my students, complained that this had little to do with how to use 'the email' and they didn't have time to learn the basics. My response was to quote the old adage: 'Give a man a fish and he will eat for a day. Teach him how to fish and he will eat for a lifetime.'

Ten years or more later, the technology has moved on dramatically but for many people, the understanding of it has not. I still teach people about ones and zeroes and indeed I have devoted an entire chapter to it in this book.

As computers have become more advanced, technology more sophisticated and the internet so pervasive, I am more convinced than ever that to survive in a world of computers, like surviving in another country,

you must take the time to understand the customs, the culture and the language.

This book is called *The Digital Migrant* because that's who it's for - people who are migrating to a digital way of life.

The Digital Migrant is written like a novel so you can read it from cover to cover. It isn't about how to write email or edit photos - there is a wealth of 'how-to' books already available for just about anything you may want to do with technology - this book is about *understanding* how the technology works in the first place.

This is the book I've always wanted to recommend to family, friends and hundreds of clients who struggle with their digital devices. Calling on my years of teaching people computers and my knack for explaining complex concepts in simple terms, I've decided to write it myself.

The Digital Migrant is not a book 'for dummies'. Why do we call people dumb just because they don't know something? How is someone even supposed to know the right questions to ask if they don't know what they don't know?

This book attempts to explain the things people don't know about computers. Where they came from. How they

think. Why they don't work sometimes. What they're capable of and what we are capable of with their help.

Although I don't shy away from getting technical, this is not a technical book. I wanted to give real, accurate explanations of how things work—not just a magical fairies version—but I also wanted to make it easy to understand and therefore I have assumed my reader knows nothing about the subject.

Any hardcore geeks reading this should note that while I have done my best to give the most accurate technical explanations possible, in places where I had to choose between simplicity and total accuracy, I chose simplicity.

Computers can be an incredibly dry subject, but I hope I've made it interesting. There should be plenty of 'aaaahhh!' moments where things that never made sense before suddenly click into place. There's also plenty of trivia tidbits that may surprise even the seasoned IT professional.

Finally, don't be worried if the technical stuff doesn't sink in. You don't have to remember the technical details, it just helps to know where they fit in the bigger picture. In my experience of teaching, seeing the bigger picture is what matters most.

If I accomplish nothing else with this book, I hope it will be to take the fear out of using computers. It's cliché I know, but they're not scary, they're just misunderstood.

My real hope of course is that you will come to find the same sense of discovery and possibility surrounding technology that I have. That I will turn you from a digital migrant into a digital native, empowered with the confidence to use technology to enrich your daily life and strengthen your connections with others.

Let's begin.

PART I

The Nitty Gritty

Chapter 1

10 Kinds of People

in which we discover the origins of computers and discuss their basic building blocks: bits, bytes and binary

Between Something and Nothing

There was silence. Heavy and expectant. The silence of someone waiting for a sound, a signal. Then it came. A flurry of high-pitched blips, some meaning intended within a pattern of seemingly random noise, and then silence again. A quick translation of the code revealed the message—'What hath God wrought?'

These were the words that marked the official opening of the telegraph line used for Morse Code in 1844. It is perhaps fitting that they were taken from a biblical verse in the Book of Numbers, for numbers are the very heart and soul of computers. Two numbers, in fact, which in turn represent an idea which has been crucial to the

evolution of humankind since we first discovered the need to count. The difference between something and nothing.

All human meaning exists somewhere in the world between something and nothing. The art of computing is simply a way of creating meaning out of both. Morse Code was developed as a way to communicate over distance. It created meaning in the form of embossed dots and dashes on a strip of paper. A 'computer' at that time referred to the person who translated the dots and dashes into everyday language. Computers as we know them today may be only a relatively recent phenomenon, but the rules by which they operate are as old as civilisation itself.

5,000 years ago, the Sumerians used two wedges pressed into a wet lump of clay to represent the idea of zero. They had become tired of dreaming up new symbols for numbers and had worked out that by making the *position* of the numbers important you could change their meaning. The same symbol for 3 in one column could mean 30, 300 or even 3,000 in a different column. The problem was how to tell which column it was meant to be in without another symbol to mark its place. A symbol was needed that literally meant nothing but historically could mean everything. And so zero was born.

Nothing is everything when it comes to computers and indeed mathematics itself. Signalling in Morse Code

using a flashlight, meaning is derived from whether there is light or no light. A code is then consulted which takes the length and sequence of this pattern of light and no light into account to produce a useful message.

A strip of paper with holes punched in it works on the same principle. If the paper was nothing but holes or had no holes in it at all it would be meaningless - it is the combination of holes and no holes that creates meaning. It is then left to a computer, whether human or machine, to decode the meaning and translate it into something useful.

It should come as no great shock that when it comes to our present numbering system, what I mean by the idea of something and nothing are the numerals 1 and 0 respectively. These two numerals in their varying patterns and sequences form a system of notation, and because it is comprised of two numbers, it is called the binary system. It is the binary system upon which the whole of digital computing technology operates.

2 + 2 = 100

Mathematics these days has largely been outsourced to computers, simply because they are so much better and faster at it, though after waiting for a few minutes while the shop assistant finds a calculator so they can work out what 10% off $100 is, you might wonder if that is such a good idea.

However, as is the case when outsourcing important work to anyone, it helps to know just enough about how to do it yourself so you can tell if your subcontractor is doing a good job.

Despite the fact computers can only understand ones and zeros, counting in binary isn't as complicated as you might think, though it can take a while for many people to wrap their heads around it. While we can all agree that 0=0 and 1=1, what does 2 look like in binary? Actually, it's 10. Here's how it works:

Our everyday system of decimal numbers is called a Base Ten system. That's because we have ten different numerals from 0–9. When we count, we start from 0 and go to 9. We then go back to 1 and add a zero. We then count to 19, and go to 2 and add a zero. This is a decimal number system (decimal meaning ten numbers), and is therefore called a Base Ten system. When we get all the

way to 99, we simply go back to 00 and put another 1 on the front to get 100.

Binary is a Base Two system and it works exactly the same way. First 0, then 1. At this point we have already exhausted all our possibilities for numerals, just like when we reach 9 in the decimal system. That means our next move is back to zero and a new column in front. Therefore, in binary, 1=1 but 2=10. After that, 3=11 and 4=100.

Another way to look at it is in columns (see the example on the following page). The column furthest to the right in the decimal system is for single numbers (also called units) from 0–9. The next column is for tens, then hundreds, thousands and so on. If you think about it, each column is 10x the column before it. 100 is 10 x 10 and 10,000 is 10 x 1,000. This is because decimal is a Base Ten system. When we write the number 16 for example, what we are really saying is there is 1 ten and 6 units (see the following table).

Decimal

10,000's	1,000's	100's	10's	1's
			1	**6**

Binary

16's	8's	4's	2's	1's
1	**0**	**0**	**0**	**0**

Since binary is a Base Two system, instead of tens, hundreds and thousands, each column represents 2x the column before it, so we end up with units, twos, fours, eights, sixteens and so on. Using the same example, that means instead of saying there is 1 ten and 6 units, we would say there is 1 sixteen and nothing left over.

In practice this means it takes more to say less in binary, but this is because computers are limited to understanding only the concept of on/off. In a punch card, each 1 would be represented by a hole and each 0 by no hole.

The table over the page shows how to count from 0–16 in binary.

Binary					Equals
16's	8's	4's	2's	1's	
0	0	0	0	0	Zero
0	0	0	0	1	One
0	0	0	1	0	Two
0	0	0	1	1	Three
0	0	1	0	0	Four
0	0	1	0	1	Five
0	0	1	1	0	Six
0	0	1	1	1	Seven
0	1	0	0	0	Eight
0	1	0	0	1	Nine
0	1	0	1	0	Ten
0	1	0	1	1	Eleven
0	1	1	0	0	Twelve
0	1	1	0	1	Thirteen
0	1	1	1	0	Fourteen
0	1	1	1	1	Fifteen
1	0	0	0	0	Sixteen

Notice how it is possible to work out what the binary number is in real terms by adding up the values of columns with a 1 in them.

Taking the idea further, we can see that the number 167 in binary looks like this:

167 in Binary

128's	64's	32's	16's	8's	4's	2's	1's
1	0	1	0	0	1	1	1

Or if we were looking at a punch card, like this:

167 in Binary

As you can see, counting in binary is not particularly difficult, but it takes a lot of effort to say very little and quickly becomes tedious. Still, next time you see one of those shirts that says 'There are 10 kinds of people, those

who understand binary and those who don't', at least you'll be in on the joke.

Character Study

So how do all those ones and zeroes translate into words on a computer screen?

By the 1960s, the Morse Code telegraph system, developed over 100 years previously, had evolved considerably. A system of Telex machines and other teleprinting devices was in full swing—transmitting news, government information and personal messages across countries and the globe. All of these devices used the binary system in the form of tape with holes punched in.

It wasn't Morse Code being used by these machines, however. As telegraphic transmission became more common, people had been looking for ways to automate the sending and receiving of telegraphs, which led to the development of new code lists, which included binary sequences not only for the usual A–Z, 0–9 and various punctuation marks, but also special sequences that would tell the machines reading them to start or stop sending,

begin a new line, start a new page or various other behaviours.

In 1963 one new such code was introduced and quickly became adopted as the standard system. Called the American Standard Code for Information Interchange, or ASCII, it assigned a **character** (a letter, number, symbol or instruction) for each binary number from 0–127, making a total of 128 possible characters. 95 of these were printable characters such as letters, numbers, punctuation marks and symbols, while the other 33 were non-printable instructions called control characters.*

If you're interested in seeing the original ASCII chart which shows the 128 different characters and their binary values, you can find it in the Appendix.

In the binary system, each one or zero is referred to as a **B**inary dig**IT**, which is in turn referred to as a **bit**. To cover all the numbers from 0–127, each character in the ASCII chart needed up to seven ones and zeros (7 bits) to write. At the time, perforated tape came in sections which allowed for up to 8 holes to be punched (8 bits), which meant there was always a spare bit left over at the beginning of each section of tape. This spare bit was

---

* For example, 7 was a control character that made the computer beep, while control character 12 caused the printer to spit out its current page and feed in a new one.

sometimes used to detect any errors in the transmission (called a parity bit) or was sometimes simply set to 0 and ignored.*

Each sequence of 8 bits read by the computer is called a **byte**, perhaps because that was the most the computer could 'chew' through at any one time. Science has unravelled many mysteries, but the sense of humour possessed by engineers is not one of them. In any case, this meant 1 character was also equal to 1 byte. A word with six letters and a full stop after it would take 7 bytes to write, while two five-letter words with a space between them would take 11 bytes to write. It has already taken tens of thousands of characters to produce the portion of this book you have read so far, which means tens of thousands of bytes. As you can see, they add up fairly quickly.

Thankfully we have names for all these big numbers and no doubt you're familiar with several of them already. For starters, 1024 bytes is called a **kilobyte**.

You may be wondering, as have many people, why, if kilo means 1000, a kilobyte is 1024 bytes. It's because of

---

* Parity bits work by counting the number of 1's in the sequence. If there is an odd number of 1's, the parity bit will also be set to 1. If there are an even number of 1's, the parity bit is set to 0. This is called 'even parity' and makes it possible for the computer to detect if there are any errors in the sequence, because if any of the bits are written or transmitted incorrectly the numbers won't match up.

the difference between binary and decimal counting. In decimal or Base 10 counting, 1000=10x10x10, or 10^3. But in binary, which is Base 2, the first power to reach the thousand mark is 2^{10} or 2x2x2x2x2x2x2x2x2x2, which equals 1024. It may not sound like much, but it's an important difference, especially when the numbers get a lot bigger.

And they do. Here is a handy table of how it all works:

Bits and Bytes

8 bits	=	1 byte	(B)
1,024 bytes	=	1 kilobyte	(KB)
1,024 kilobytes	=	1 megabyte	(MB)
1,024 megabytes	=	1 gigabyte	(GB)
1,024 gigabytes	=	1 terabyte	(TB)
1,024 terabytes	=	1 petabyte	(PB)

It goes on, but that should do to get you started. As for those 24s I mentioned earlier, consider this. 1 gigabyte is 1,073,741,824 bytes. That's a difference of over 73 million bytes compared to calculating using 1000 instead of 1024.

If you've ever looked at a brand new 250 gigabyte hard drive, only to discover the computer says it has a capacity of 232.83GB, you'll have noticed this difference. It's because of exactly the phenomenon we just talked about. Manufacturers of hard drives use multiples of 1000 bytes when calculating the capacity of their products, whereas most computer operating systems use the binary method of 1024, which explains the discrepancy. It's actually quite a heated debate in certain circles, but if that's what passes for entertainment for them, you can be fairly sure those are not circles you want to be part of. Let's move on.

ASCII is still in use today, although over time it has been extended to include more possibilities such as mathematical symbols and foreign language characters. Eventually it began to be phased out in favour of Unicode, a more flexible character system that allows for just about every conceivable character in any language.

Because perforated tape was expensive, such advances in the early days of computers were limited and slow, but that was before computer technology took a giant leap forward with the advent of the microprocessor.

Essentially a complex collection of transistors and connectors built into a piece of silicon, the microprocessor delivered a way to process information much faster and

more efficiently than ever before. In 1974, after a few earlier designs had already made the rounds, an electronics company based in Santa Clara, California, released a new microprocessor called the 8080 which would lay the foundation for computers as we know them today. That company was Intel, and, after its technological breakthrough, the Santa Clara region in California soon became known as Silicon Valley.

Just like the perforated tape, Intel's 8080 microprocessor was capable of handling 8 bits at a time. What made it such an amazing breakthrough was the sheer speed with which it could process these bytes of information. The tape devices at the time were handling around 9 or 10 cycles of 8-bit tape per second, an impressive feat in itself. But Intel's 8-bit microprocessor could run around 2 million cycles every second, thus signalling the end of analogue computing and throwing open the door to the digital computing revolution.

Free Samples

With their ability to work through millions of ones and zeros per second, microprocessors opened up new possibilities for putting the binary system to use. We've

already seen how ones and zeros are used to produce text, but what about the sounds, images and movement we experience on computers and other digital devices? These are all created using the binary system too.

Everything in nature, no matter how complicated, can be traced back to its component atoms. Although computers are machines, they share a similar design philosophy with nature in that big things are made up of lots of little things put together in different ways.

From these simple building blocks there are infinite possibilities. Creation is a matter of finding patterns and sequences that work. Digital technology works by converting information into binary data and then reconstructing it into text, images and other forms of information that can be understood by people.

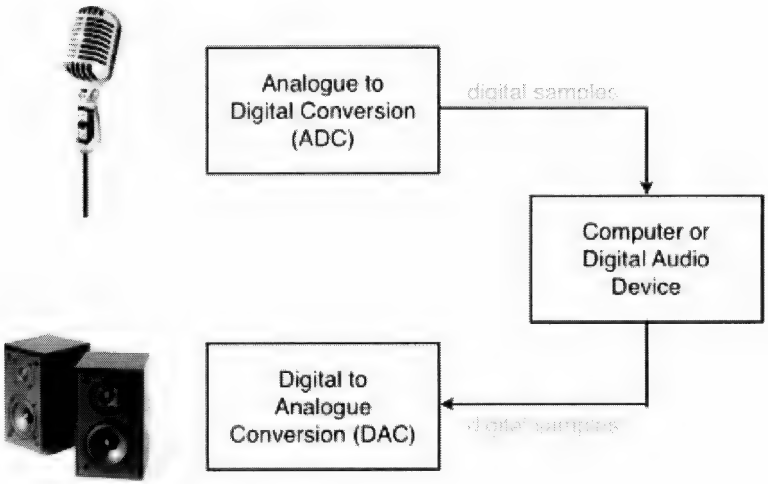
I will try to explain how this occurs in the simplest language I can while striving to be as accurate as possible. If it gets a little technical don't be too concerned. It isn't vital to understand the inner workings of a computer to use one, but it's surprisingly helpful and does wonders for self-confidence.

Consider sound, which is caused by vibrations through the air or other substances at frequencies which we can hear. Sound is analogue—it is a continuous signal. A music CD is digital—it is a sequence of numbers. How is

it possible to take sound waves out of the air and translate them to a piece of plastic and metal that plays music when you put it in a CD player?

The answer is a process called **sampling**. To explain in simple terms—a microphone listens to the continuous sound waves (called the waveform) and records the audio frequency at regular intervals, creating a flow of numbered values, which, if played back in the right order through a speaker, will reproduce the original sound. The microphone is literally taking samples of the sound. The number of samples it takes each second is called the **sampling rate**. Although the result will never be as pure as the original waveform, if enough samples are taken, the difference should be imperceptible to the human ear.

The diagram over the page shows the digital audio process:


Analogue to Digital Conversion (ADC)
digital samples
Computer or Digital Audio Device
Digital to Analogue Conversion (DAC)

The numbered values recorded during the sampling process, are, needless to say, written in binary. *

Samples on a CD are represented by a series of bumps burnt into the disc by a laser, called pits and lands. When the disc is playing, the laser runs along the track, reflecting the light differently when it reaches each pit. This is how a CD player understands binary code.

---

* Interestingly, to calculate the number of samples necessary to make the digital audio recording imperceptible from the original, a formula called the '*Nyquist-Shannon sampling theorem*' is used. It states that the ideal number of samples should be twice as many as the total range of frequencies being sampled. Since humans can hear frequencies in the range 20hz - 20,000hz, the ideal sampling rate for audio recording should be around 40,000 samples per second, or 40khz. In fact, the sampling rate of compact discs is 44.1khz.

While each second of digital audio is comprised of thousands of tiny parts called samples, each digital image is comprised of thousands or even millions of coloured dots called **pic**ture **el**ement**s**, or **pixels**. Each pixel is a particular colour, and, like the ASCII text system, each colour is stored as a corresponding number, written in binary. The computer reads the binary code for each pixel, translates it into the correct colour and then draws each pixel on the screen, creating the complete image.

Computer screens produce colour by combining red, blue and green light at different levels of intensity. This is called RGB colour. RGB is called an additive colour model, because when red, green and blue light are all added together at their highest intensity, the result is white.

Printers use a different colour model called CMYK in which colours are produced by mixing cyan, magenta, yellow and black inks. Whereas in RGB, all the colours mixed together create white, in CMYK, cyan, magenta and yellow mixed together create black. CMYK is called a subtractive colour model because you start with a white page and subtract the white by mixing the other colours together on top of it. This difference in screen and print colour models explains why the colours of printed images don't always look the way they do on screen.

In the early days of computers, only one screen colour could be produced, usually white or green. Thus each pixel was represented in binary by a single 1 or 0. If the pixel was set to 1, the computer would draw colour there. If it was set to 0, it would stay black. Later there would be 2-bit colour, two ones or zeros together for a combination of up to 4 colours. Programmers could choose from two amazing colour palettes - cyan, magenta, black and white, or red, green brown and black. Game designers went wild with the possibilities! (you can see some examples of early colour models in action at www.thedigitalmigrant.com)

As computing power and graphics processing improved, so too did the range of colour options. Today, the standard for colour reproduction is called 24-bit or 'true colour'. It is produced by combining 256 possible shades of red with 256 shades of green and 256 shades of blue, making 16,777,216 colour combinations, enough to make an image appear natural. The 256 shades are represented in binary using the numbers 0-255, each taking 8 bits to write. Combining the individual 8-bit values of red, green and blue explains how we get 24-bit colour.

A digital camera samples the light reflecting off sensors in the lens and converts it to RGB values. A 5-megapixel camera records images using 5 million pixels. Each pixel takes 24 bits to produce. That means a single 5-

megapixel image can take up to 15,000,000 bytes of information to produce, or 14.31Mb.

One second of video footage is made up of 25 or more still images. With that amount of computing power required for a single task, it begins to become clear how much work the computer is doing behind the scenes.

Strength in Numbers

All human meaning exists somewhere in the world between something and nothing. Throughout our history we have sought to capture meaning and record it, for ourselves, or for our future selves. In doing so with ones and zeros—the literal embodiment of something and nothing—there is a kind of poetry at work. We can capture anything, no matter how subtle or complicated, if we only have enough numbers. And all that we do capture, whether magical or mundane, is made from the same dust. It is all, in the end, just bits.

And yet there is strength in numbers. These numbers are alarmingly precise and surprisingly flexible. Most importantly, they are reproducible. The binary system offers a way to record information that can be reproduced

quickly, easily and flawlessly in its entirety. This is an incredible achievement in itself. But in addition, the binary system, by its very nature, works in such a way that any individual piece of that information, no matter how small, can be identified, extracted or altered, and that is what makes it simply mind-blowing.

It's amazing what we take for granted. Typists, before computers were used as word processors, would have been incredulous if they were told that whole chunks of text could be edited and moved around without having to be retyped. Before the printing press, only the very privileged few ever had the chance to read a book. Can we imagine a world with no electricity? It was so once, and not so long ago. It's hard to imagine life without cars, cameras, television and computers, but these are all technologies that, for some, have appeared within living memory. Those born today will never know a world without the technology this book explores.

Technology changes everything. Once something has been invented it stays invented. Once something enters consciousness, it cannot be removed without the removal of consciousness itself. Technology creates empires and destroys others. Wars are fought for it, with it and prevented by it.

Technology is inescapable. It pervades every aspect of our lives. The course of history and the history of technology are inextricably linked. As for the future, renowned computer scientist Alan Kay once said the best way to predict the future is to invent it.

Binary information, at microprocessor speeds, is the essence of digital technology. Think for one more moment why digital technology is so mind-blowing. Whatever we experience digitally, whether it be words, pictures, sounds, movement, none of it is really solid. What we are experiencing is an endless stream of numbers, silently transformed in myriad ways, perfect patterns from mathematical noise to create meaning that is real, just for a moment, before collapsing back into infinity. Creation and destruction too quick for the eye, ready to be recalled again at any time, in any form. Wrought of creation and nothingness in their purest forms.

Strength in numbers indeed.

Chapter 2

Ins and Outs

in which we learn how a computer thinks and in which memory, processing power and operating systems are explained

I Think, Therefore I Am

It is evening and a group of men are seated around a table deep in discussion. They are debating whether a machine can be made to act like the human brain. After sitting quietly for a while, a man at the end of the table interrupts:

"The answer is to make a universal machine. One that is capable of turning itself into any other machine."

The rest of the men shuffle uncomfortably. They always felt a little uneasy around this man. Someone asks:

"How would it know what to do? Is it possible to give a machine purpose?"

The man at the end of the table responded, "It can be done with trial and error. Purpose is the use of previous combinations plus trial and error."

"But how would it learn, how would it choose what to do?"

"Random operation can be made to become regular after a certain prevailing tendency has shown itself." There were a few murmurs among those present. Who invited this man anyway? But he continued:

"I am thinking of the kind of machine which takes problems as objectives, and the rules by which it deals with the problems are different from the objective. For instance, if I do an addition sum on the blackboard I can do it either by consciously working towards the solution or by following a routine habitual method. I get the same result either way, but they are two different things and they should be kept separate."

At the other end of the table, one of the other men leaned forward and said:

"The vital difference seems to be that a machine is not conscious. Every aspect of a machine can be fully specified, while the mind cannot."

The young man replied immediately, "The mind is only said to be unspecifiable because it has not yet been

specified. A machine may be full of incompatible choices, but when it gets a contradictory result, there is then a mechanism to go back and look at things which led to the contradiction."

"But surely this is an argument against the machine—humans don't do this kind of thing do they?"

"Yes, mathematicians do."

The men whisper among themselves again. Someone mutters: "Are mathematicians human?"

This discussion took place in 1949 in Manchester, England. The young man was Alan Turing, already well-known for the code-breaking machines and formulas he developed during World War II. At the time, any machine designed to do computing had to be wired and set up in a certain way. If you wanted the computer to perform a different task, it had to be painstakingly redesigned and then physically rebuilt, something which could take weeks. Turing continually emphasised the importance of what he called a 'universal computing machine', a computer that could be used for anything. He had written about it as early as 1936 and had done much research and work towards the concept, including the making of a prototype. The universal computing machine wouldn't need to be physically rebuilt to change tasks, it would simply rearrange itself internally.

Alan Turing is today widely considered to be the father of modern computing, as well as a pioneer in the field of artificial intelligence. In 1948 he began writing a chess program, but by 1952 there was still no computer in existence that was powerful enough to run it, so Turing set up a game where he played as the computer himself, following the program's logic. It took about half an hour for each move. The program lost to its human opponent, but it inspired many in the field of computer science to further the concept.

Despite Turing's many achievements however, it is another man who is given credit for designing the computing systems we use today. Hungarian-American mathematician John von Neumann took Turing's universal machine idea and made it a reality. Not only was he responsible for this major technological breakthrough, but John von Neumann also made significant contributions to quantum physics, economic game theory and was a member of the Manhattan Project that developed the atomic bomb.

The von Neumann architecture, as it is known, is fundamentally simple. Instead of a computer processor that follows a fixed set of instructions, the processor fetches instructions and data held in a separate store which can be reprogrammed easily as required. This store of information is referred to as the computer's **memory**.

When we begin to use terms like memory in relation to machines, we unavoidably draw comparisons between humans and computers. Can a computer think? Can it ever be conscious? What will happen if it does? These questions have been explored time and again throughout our culture in books, movies and philosophical debates.

Alan Turing foresaw many of the questions and concerns arising from artificial intelligence and published his thoughts on them in his now famous 1950 paper 'Computing Machinery and Intelligence'. In doing so, he chose not to explore the question 'Do machines think?' by following the traditional method of first defining what a machine is and what intelligence is. Instead, Turing posed a different question—'Can machines do what we (as thinking entities) can do?' In this approach, the debate about consciousness is secondary, the question is whether computers can fulfil the role of a human in a given situation.

To demonstrate his way of thinking, Turing established what is now perhaps his most famous legacy—the Turing Test. The test was a variation on a popular party game in which a man and a woman would go into separate rooms and guests would have to try to guess who was who by asking and receiving a series of typed questions. Both the man and the woman would pretend to

be each other and it was up to the guests to gauge from the responses who was the real thing.

In the Turing Test, one of the participants is human and the other is a computer. The interrogator is tasked with deciding which is the human and which is the computer by asking each a series of questions of his or her own choosing. The questions and the responses must be typed. If the interrogator can't correctly pick the computer's answers, the computer is assumed to be as intelligent as a human, at least for the purposes of the test.

To date, no computers have passed the Turing Test, though a couple have come close. However, computers have become much better at chess. In 1997 IBM's Deep Blue chess computer beat reigning world champion Gary Kasparov in a well-documented tournament match.

I know my computer gives me a solid thrashing at chess whenever I play against it, even when I follow all of its hints, suggesting at least to me that computers are capable not only of intelligence but of a certain degree of malevolence as well.

Computers are dangerous. They have their own languages and they talk among themselves. They take our jobs and undermine our society. So goes one theory anyway. As with most prejudices it's due to ignorance, which in turn can be combated with understanding and

familiarity. Computers can certainly adapt, interact, speak and interpret, which makes them intelligent, at least for practical purposes. We know they think. But do we understand *how* they think?

A Box of Memories

Although a computer can 'think' about millions of ideas each second, it is important to understand that when you break it right down, it can only think about one thing at a time. Many women may conclude that this is because a man designed it.

This fundamental limitation causes problems even and especially with today's computers, because even though the brain of the computer, the **Central Processing Unit** (**CPU**) can operate at blistering speeds, other parts of the computer can't keep up, causing bottlenecks throughout the system.

The Colosseum, the famous ancient stadium in Rome, could seat 50,000 people. It was designed in such a way that the entire ampitheatre could be filled or emptied within minutes, thanks to a clever system of pathways called *vomitoria* (from the Latin '*vom*' meaning rapid

discharge.) If the Colosseum is the gold standard of architecture for practical and efficient use, the modern computer using the von Neumann architecture is the equivalent of a suburban shopping centre carpark during the week before Christmas.

The CPU handles instructions in a four-stage cycle. First, it fetches the next string of numbers in the queue. It then takes a look at the numbers to see what they mean and decide what to do with them. Once it has decided what to do, it carries out whatever needs to be done and then finally sends the results back out to be stored. Then it fetches the next bit of information and the cycle begins again. This is called an **instruction cycle**. The four steps are usually referred to as **fetch**, **decode**, **execute** and **store**.

The number of instruction cycles the CPU can process each second is referred to as its **clock speed**. A CPU with a clock speed of 2GHz indicates that it can process two billion instruction cycles per second. Yes, two *billion*.

Let's consider that for a moment. Imagine counting numbers out loud, starting at 1. Counting at the rate of one number per second, how long would it take to reach two billion? A little over 63 years, in fact. If you had one thought each second for the next 63 years, you'd match what a 2GHz CPU can manage in a single second.

You might expect then that the higher the processor's clock speed was, the faster the computer would be. That's certainly how computers have traditionally been marketed —having a 'fast processor' is often considered the be all and end all of computer performance. But this is a little misleading.

In practical terms, a computer is only as fast as its slowest component. When we think 'fast', what we usually mean is 'responsive'.

This is also misleading. In reality, a computer is a complicated network of different components with different capabilities and purposes all operating under the control of another complicated system of language with its own idiosyncrasies. But fear not, by the end of this chapter you will have a good understanding of the physical system, called **hardware**, and the operating system which controls it, called **software**.

As mentioned earlier, the CPU containing the microprocessor is the brain of the computer. It is where all the calculation, logic and decision-making happens and it has the fastest operating capability of the whole computer. The CPU both takes its instructions *from* and delivers the results *to* a store of information called the computer's memory. It may help to think of the memory as a turnstile with information constantly passing in and out. The

memory also has a clock speed which is typically much lower than that of the CPU. This is the primary cause of the information bottleneck, since all the information going to and from the CPU must pass through the memory. As you might imagine, there has always been much focus on experimenting with different types of memory technology to try to increase efficiency and clear the information bottleneck.

Memory comes in many different forms. When considering the memory that connects directly to the CPU, it is essential for efficiency that the CPU be able to write to and retrieve information from any part of the memory at any time—at random as it were. Thus it is called **Random Access Memory**, or **RAM**. However, the RAM used in most computers stores its information electrically, which means when there is no electricity there is no way to hold the information. This sort of memory that requires power is called **volatile memory**.

A computer that loses all its information whenever it's switched off or as soon as there's a power cut isn't much use, so a secondary non-volatile memory must be installed to store the data when it is not in active use by the main memory. There have been many different types of non-volatile memory storage over the short history of computers, notably the floppy disk, hard disk, CD, DVD and Flash memory card.

Both the floppy disk and hard disk work on the principle of magnetism. In simple terms, a circular disc coated in magnetic film sits on a spindle and spins very fast. A needle sits very close to the spinning disc but not quite touching it. To write information to the disk, the needle changes the magnetism in the disc as it spins by, and reads the magnetism back when it needs to retrieve information.

Looking at today's hard disks, magnetic disk technology has developed to allow very large amounts of data to be stored, but has significant drawbacks. Firstly, because it has moving parts, it is fragile and susceptible to dirt or damage. Most hard disks spin their magnetic platters 7,200 times a minute—if the needle ends up touching the surface of the disk for any reason it can destroy large amounts of data very quickly. Many people who have experienced hard disk failure and loss of important data know how frustrating this can be. Secondly, because the platters are magnetic, they are sensitive to interference from other magnetic devices. Sticking a floppy disk to a fridge with a magnet is the quickest way to remove all hope of ever retrieving what was on it.

Perhaps the most significant drawback of magnetic disk technology is that it further slows down the information flow. Just as the computer's RAM isn't as fast

as its CPU, the hard disk isn't as fast as the RAM. Whenever the computer needs to swap data in and out of permanent storage into active use, it takes time. So what's the alternative? Not CDs or DVDs, which are even slower at reading data from than hard disks. It takes far longer to record information onto a CD or DVD than a hard disk and recording can only be done in huge chunks and not one byte at a time, thus making the 'random access' idea unworkable.

The most promising alternative appears to be **Flash memory**, a random access non-volatile technology much faster, quieter and more rugged than a hard disk because it has no moving parts. Flash memory has been around for a while, developed at Toshiba in 1980. It got its name because at that time you had to clear the whole memory and start from scratch if you wanted to change any information stored on it, a process that reminded its creator of a camera flash. Happily, that limitation no longer exists—varieties of Flash memory can be found in digital cameras, mobile phones, portable music players and gaming devices. It is widely expected that Flash memory, also called 'solid state memory' will eventually replace the hard disk as the primary storage method in computers as prices fall and technology improves.

Nevertheless, despite the varying ways in which they operate, all types of memory perform essentially the same

function—to act as a store from which the CPU can retrieve and deliver information as quickly and efficiently as possible. But what sort of information? Where does it come from and where does it go?

On the Buses

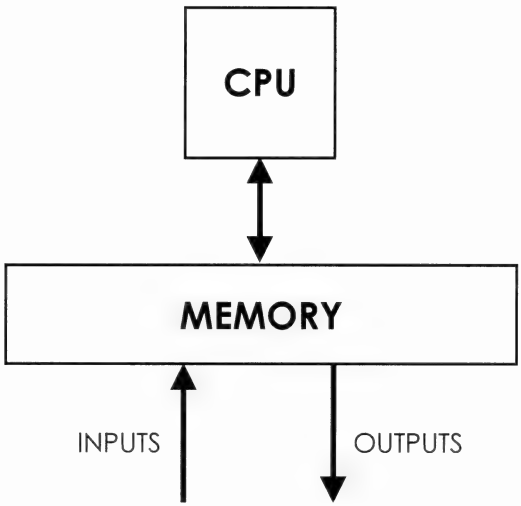
Like the hordes of suited businesspeople on a busy morning in a large city, all the bits of information bustling around the computer are on a mission, travelling from one place to another armed with purpose, and, from an outsider's perspective, possibly an inflated sense of self-importance.

From the perspective of the CPU—head office if you will—every bit of information is either a request issued by another department to be looked at and actioned as soon as possible, or an edict sent hurtling through communications channels to be carried out immediately by an underling. At all times while the computer is running, information is either coming in or going out.

To persist with the corporate metaphor, some departments are set up entirely to gather information and send it to head office. In a computer, such departments

are called **inputs**. Other departments exist solely to receive information from head office and present it to the customer. These are called **outputs**. Another group's role is to facilitate the transfer of information from one department to another and to keep reports on performance and efficiency, but apart from that they don't really do very much except get in the way. These are called middle management. Well maybe not that last one, but you get the idea.

Some examples of inputs include the keyboard and mouse, microphones, cameras and scanners. These are all devices that give the computer something to think about. Some examples of outputs include audio speakers, printers and of course the computer screen. These are all devices that present the results of the computer's thinking. Some parts of the computer act as both inputs and outputs, such as the memory. See the diagram over the page for a visual representation:



For an example of inputs and outputs in action, consider what happens when you type the letter 'A' on the keyboard. The computer takes the input from the keyboard, passes it up through the memory and into the CPU. The CPU works out that you have requested the letter 'A', finds out what it should look like, then passes the instruction back through the memory and outputs it to the screen.

The same applies to moving the mouse. As you move the mouse, you are inputting information to the computer which says how far you have moved and in which direction. This information is passed up through the memory to the CPU, which then outputs an instruction to move the mouse pointer on the screen accordingly. From the point of view of the person using the computer, this

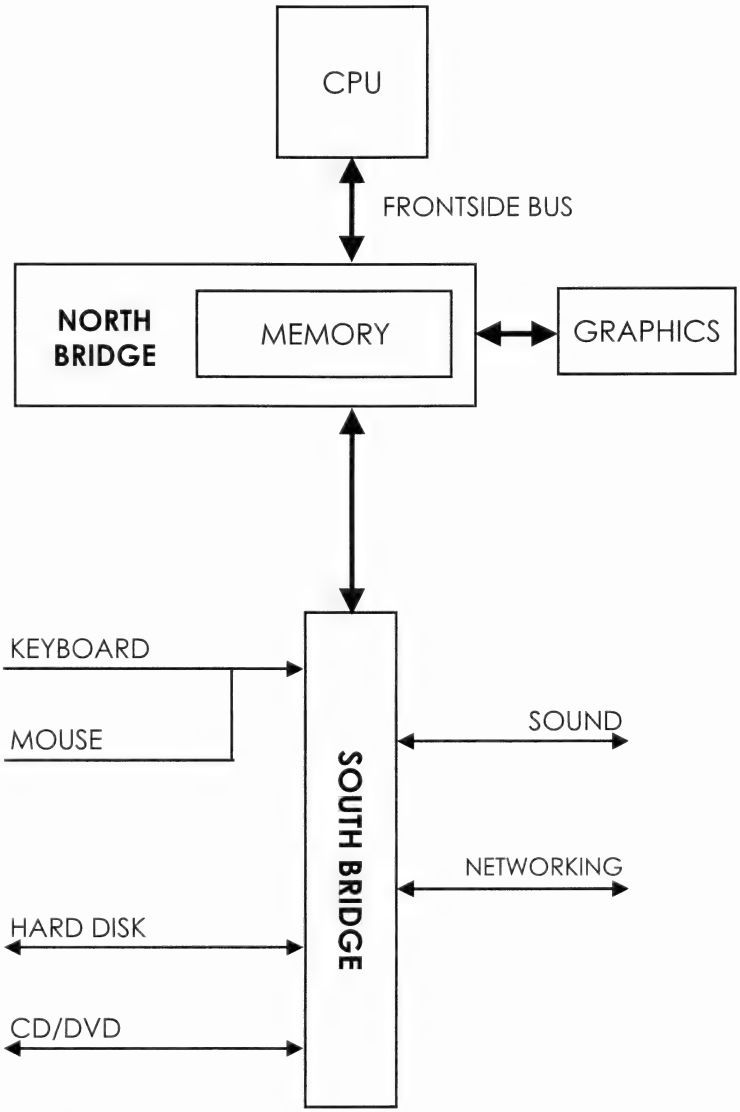
happens immediately, a fact which belies the complexity of what is really taking place.

All information travelling in and out of the CPU must pass through the memory. To minimise traffic congestion, the memory enjoys exclusive access to the CPU, with which it communicates along a dedicated circuit called a **bus** (it seems when it comes to computers, even the busy and powerful have to take the bus.) In fact, there are buses all over the system, which is how information travels from one part of the computer to another. They follow different routes and run at different speeds, but they all link up eventually. The bus between the memory and the CPU is the fastest and most important. It is usually referred to as the **frontside bus** and is depended on heavily for the overall speed and performance of the computer.

Of the outputs, the computer screen is the most important and requires the most work to operate. Because of this, the memory also has an exclusive high-speed connection with the graphics hardware which controls the screen. Together, this cosy club comprising the CPU, memory and graphics controller resides in what could be thought of as the flashy end of town where the best hotels and the fanciest restaurants are. In most of today's computers this part of town is called the **North Bridge**.

At the other end of town you'll find there's still plenty of activity but things run at a more leisurely pace. The buses are less frequent and more congested and there are more stops along the way. Such is life in the **South Bridge**. Nevertheless, the intrepid traveller will be able to find their way to all the other components of the computer from here.

The diagram on the opposite page gives a basic outline of the layout of a modern computer.



As you can see, the CPU sits by itself, connected to the memory via the high-speed frontside bus. The memory also connects to the graphics controller which

tells the screen what to display. The North Bridge is connected to the South Bridge by another dedicated bus, and the rest of the computer's components are attached to the South Bridge. This representation is a simplified version of the real thing and is missing some of the more complicated components, but the purpose of this diagram is simply so you can see how the pieces of the computer connect together and how inputs and outputs flow through the system.

Layers of Abstraction

If you are planning a holiday you are probably going to need a travel agent. Of course you could try organising that trip around Europe by yourself, booking the flights, accommodation, insurance, transport and activities and saving yourself the agent's fee. That is assuming you knew who to call, were able to get the best rates, knew how to quickly calculate travel times in different time zones and could speak each language fluently. Similarly, you could order your next meal at a restaurant by going out the back and talking directly to the chef. It might save you tipping the waiter but it may well be the last time you eat there. Next time you go to see someone at their office without an

appointment, you could just walk in and ignore the receptionist, but that might get you thrown out of the building.

The point is, when you're dealing with complex systems, there is inevitably someone or something that acts as a host, whether they be a travel agent, a waiter or a receptionist. For computers, the role of host is played by the **operating system** (**OS**). If you want to be able to use a computer without a thorough working knowledge of binary and a background in electrical engineering, you'll probably want an operating system to do all the talking to the components and deal with all those billions of binary digits for you.

An operating system itself has many functions and different departments, but the most important is the **kernel**. The kernel of an operating system is like the CPU's chief of staff. Of all the seething mass of data flowing around and being received, it has to prioritise the decisions to be made by the busy CPU. It decides whose messages are urgent, who can wait and who are simply troublemakers. It is tasked with keeping the CPU on schedule and will not hesitate in forcibly bringing appointments to a close if they drag on too long or stray too far off topic.

The job of kernel became necessary as computers evolved to be the universal machines Alan Turing had envisioned. Just as a restaurant with only one customer does not need a waiter, computers that were built with only one application in mind had no need for an operating system. It gradually became clear however that just as an entertainment hall is more viable when it can be used for concerts, sports *and* conferences, computers would be more useful and much cheaper to produce if they could be used to perform many different tasks. However, such a move would require a versatile design and fairly clever management to succeed in practice.

To achieve greatest efficiency in management, most industries have a system of tiers, each with their own sets of policy and procedure. Each tier usually has a relationship with the level directly above and below it, and communication passes up and down through one tier at a time. The sales assistant in a department store may know what the CEO looks like but would be surprised and probably a little unsettled if they came to talk to him or her directly. Of course, we are talking about humans here, which means there are all sorts of emotional behaviours which come into play and invariably make a mockery of the tiered management system, frequently bringing its value into question. But for computers, living in a world of

pure mathematics and free as it were of emotional attachments, the tiered management system makes sense.

The bottom tier is the hardware, upon which everything depends and without which nothing would work. Ultimately the hardware is commanded at the top tier by way of all the flashy graphics, menus and buttons we see which make up the software, but a lot goes on in between.

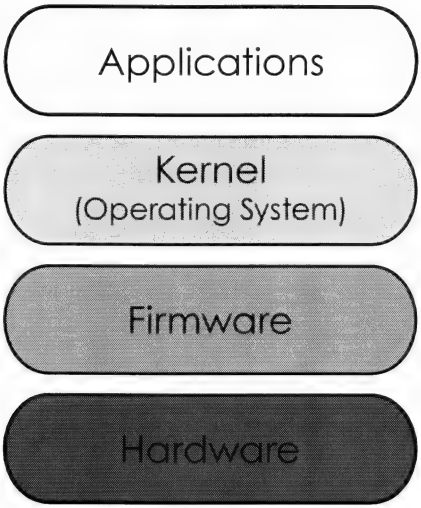
The second-bottom tier directly supervises the hardware, each piece of which has its own unique set of rules and working conditions which vary between models and manufacturers. The highly specific set of instructions which controls each hardware component is called **firmware**. That's because it's between hardware and software. Get it? This is another example of why engineers are not known for their sense of humour.

We have already mentioned the kernel, which supervises how the tiers below spend their time and prioritises the processes that need to be done. These processes are like the different functions of the entertainment hall mentioned earlier—concerts, sports and conferences for example. The hall has to be rearranged for each, which takes time, so it makes sense to book as many of one type of event together as possible to minimise disruption and maximise availability. This

sort of management is what the kernel concerns itself with, where the events are the various programs the computer is running (also called applications or processes), such as word processor, internet browser, email and music jukebox software.

Another benefit of the tiered management system is that each tier functions independently of the others and consequently can be modified or replaced independently. A company can replace its board of directors without it directly affecting the frontline staff and vice versa. This is because the positions everybody in the company holds are abstract. Each position has its own requirements, but can be filled by anyone with the appropriate ability. Even in the worst case scenario that a company's entire staff were lost in a disaster, the abstract nature of its management structure means those positions could eventually be refilled.

In computer science this idea is just as important, but management tiers are instead called **layers of abstraction**. The abstraction layer model demands certain hardware be present and that it be arranged in a particular way, but that hardware could be supplied by a variety of different manufacturers in a variety of different models. The kernel needs to be able to work with many different CPUs, just as each CPU needs to work with many different kernels.



Layers of Abstraction

When you consider the mind-boggling array of add-on devices used with computers such as printers, scanners, cameras, headphones, phones, modems, games controllers, graphics tablets, barcode readers and so on, it is unrealistic to expect the kernel to be able to instantly recognise and converse fluently with every piece of hardware it meets. Many of these devices are small computers in their own right. This has led to the need for a separate hardware abstraction layer within the kernel.

Like a diplomatic relationship between foreign countries, a deal was negotiated as follows—the kernel would provide a list of topics that could be discussed as well as acceptable requests that could be granted, while the hardware manufacturer would provide a list detailing

which of these topics and requests it was interested in for the device in question. The information provided by the hardware manufacturer would then be used to 'drive' the relationship between the two parties. Hence this class of information is called the **device drivers**.

Device drivers, like trade and immigration controls, can have a significant effect on prosperity and stability. Few or no restrictions on trade can lead to a huge number of trading partners but little quality control. Tight controls can mean increased quality and scrutiny but may stifle interest. The way a kernel handles this and many other issues has a great bearing on the quality and appeal of different kernels and operating systems, as you will see in Chapter 4.

Going All Gooey

For a long time, the realm of computers was confined to mathematicians and electronics enthusiasts, largely due to cost, a steep learning curve and the fact that not many people enjoy mucking about with wires and pliers.

The concept of an operating system provided a buffer which saved the end user from having to directly program

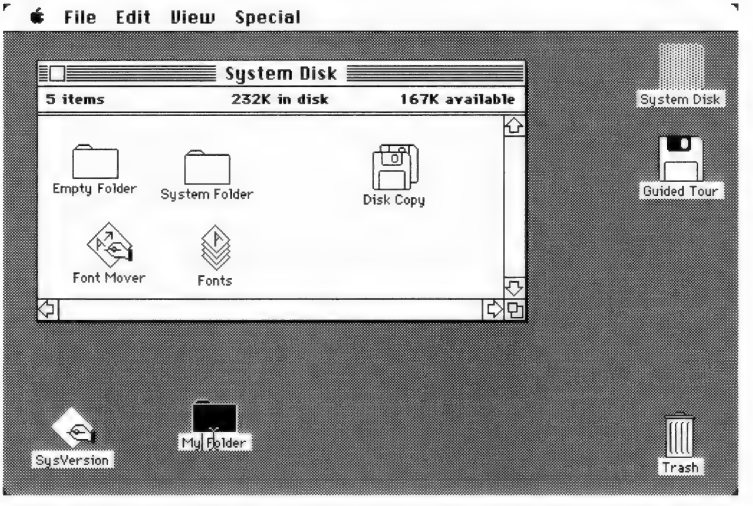
the hardware. Having an operating system to do most of the programming meant, in effect, that the computer could program itself using broader commands given by the user. However, the means of interacting with the operating system was still limited to typing arcane combinations of almost-words onto a blank screen, which still required considerable technical skill as well as zen-like calm and patience.

During the 1970s, engineers at Xerox's Palo Alto Research Centre (PARC) in California were working on the creation of a visual way to interact with the computer. They developed a system which represented various typed commands using an array of pictures and buttons which could be navigated using a pointing device. The system was dubbed the **Graphical User Interface** (**GUI**). Its chief architect was Alan Kay, who famously said the best way to predict the future is to invent it.

The future arrived in 1984, not in the manner George Orwell had foreshadowed, but rather the opposite. Far from being a society deprived of all information but that meted out by Big Brother, the arrival of the personal computer marked the start of a new age where individuals had access to tools and information never before available.

Working with and borrowing from Xerox PARC, Apple had developed a new computer priced for personal

users, sporting a newly designed GUI, a mouse and a handle so the computer could be carried. It was launched amidst much fanfare in 1984 and became the first commercially successful computer to use a graphical user interface. It was called Macintosh.



The 1984 Macintosh GUI

Despite significant increases in capability and features, not much has changed in the basic concept of the graphical user interface since it was first introduced. The prevailing visual metaphor is still the desktop screen with applications in floating windows controlled by a mouse pointer.

Another concept that has thus far withstood the test of time is the storage metaphor of **files** and **folders**. Because the information stored in a computer is all ones and zeros, without some sort of filing system it would be impossible to know where one section of data stopped and another began. This becomes even more apparent when computers are running more than one application. In an endless sea of ones and zeros, how do you know what is a treasured family photo and what is a report on eco-friendly irrigation?

The idea of a computer 'file' is simply to take a chunk of information and keep it together with a little note that says what it is. A file contains both the data itself and some extra information such as its name, when it was created, who created it and which application it is to be used with. This extra information is called **metadata**. A file could be a document, a piece of music, a photo, part of an application or hundreds of other possibilities. To keep things tidy, files are stored in folders, which can in turn be stored in bigger folders and so on.

The operating system invariably includes an application which allows the user to browse this system of files and folders and move things around as they desire. The Macintosh GUI was designed to give the impression of a virtual office, hence the desktop and filing metaphor. Today's computers however have stretched this concept to

breaking point with literally thousands of folders and millions of files, making it difficult to find things quickly or know where everything is.

One solution to this problem has been to use the metadata of each file and its contents to create an index of the entire file system. Then, instead of browsing through folders, the user types what they are looking for into a box and the computer searches the index and delivers the results almost immediately. This method allows the user to locate a file without necessarily knowing its name or location, since the index contains everything about each file including its contents. An effective example of this technique in action is the 'Spotlight' feature in Apple's Mac OS X operating system, which practically negates the need to think about files and folders at all, since anything can be searched for and found almost instantly.

Features like Spotlight are only possible in a well structured operating system. To deliver fast and accurate results, Spotlight must keep an up-to-the-minute index of every file and its contents in the entire system. This means Spotlight must have close communications with both the kernel and each application in use by the computer. It achieves this by using an **Application Programming Interface** (**API**), which is basically a two-way phrasebook that allows other parts of the computer to speak the same language as Spotlight. Using the Spotlight

API, software developers can program their own applications to work with Spotlight to provide a better experience for the end user.

An advanced operating system like Apple's Mac OS X has many APIs that provide fast and flexible functionality to software developers and end users. Among these 'core' services are APIs for audio, image, video and animation, providing a suite of built-in multimedia features that can be freely used and shared by any application. This is one of the reasons Apple computers have a reputation for being well-suited to creative industries.

The sophistication and ubiquity of modern personal computers gives the illusion that they are simple enough for anyone to use. In reality they have become more complicated than ever. Where the operating system could once have been considered a single entity, almost a simple application in itself, now it is a complex collection of interconnected subsections in its own right.

Everything about a computer is engineered to achieve efficiency and simplicity for the end user. But occasionally when things stray a little too far off the beaten path, the user is confronted with a sudden glimpse of the computer's inner workings and the mirage of simple servility vanishes. Hence, like lion taming, it is a good idea

to understand at least a little of what one is dealing with before one pokes and prods it too much.

A Mind of its Own

Can machines do what we as thinking entities can do? Looking at today's computerised world, it would be hard to conclude otherwise. Yet do machines think? The answer really depends on what is meant by thinking.

It is easy to form the view that a computer does think, although not like a human nor even like an individual. It could be said that a computer thinks more like a city. It is full of individual and often incompatible ideas, self-focussed thoughts running in all manner of ways and yet somehow managing to flow together in one direction when required. It is vaguely conscious, though not of itself, but rather as a product of the life which courses through it. When there is no life, there is no thought. In that case a computer may be more like an ant colony or a beehive—more than the sum of its parts. Indeed, its parts have little interest in or awareness of themselves beyond their capacity and desire to serve the whole. There's a word for this phenomenon—*gestalt*.

It is perhaps no coincidence then that the computer was not designed by an individual. Neither has its history nor its fate at any time rested in the hands of an individual, though there have been many who have influenced computer technology and many more who have been influenced by it.

Technology is usually considered from the perspective of the effect it has on society, or the effect society has on technology. Rarely is it considered the effect that technology has upon itself.

If a computer thinks like a beehive, what happens if the bees from one hive cross over to another? What happens if a group of computers get together and share each other's thoughts? As it turns out, something very interesting indeed.

Chapter 3

The Cloud

in which we explore the nature of the internet and learn how search engines work

The Sum of All Knowledge

It was too late. By the time anybody knew what was happening, there was nothing they could do. Their faces lit by the flames, they could only watch as it all went up in smoke. Everything they had worked so hard for, everything they had tried to treasure and protect—gone. Ash and smoke filled the sky. What had existed only moments ago was now simply a cloud of particles, scattered and meaningless. The great library was no more.

The Library of Alexandria was the largest library in the ancient world. It was founded around 300BC in Egypt with the goal of being the ultimate store of all human knowledge. A great research facility funded and maintained by the Egyptian Royal Family, the Library

contained works on mathematics, physics, natural science and astronomy, all written on papyrus scrolls. Copies of certain works would be painstakingly made for royalty or those very rich scholars who could afford it.

Eventually the great library was destroyed. There is still much debate as to when, why and by whom it was destroyed, but perhaps we should not be so surprised by this. If the ultimate store of recorded human knowledge goes up in smoke, one would expect there wouldn't be much record of it!

Knowledge became much easier to record in the mid-1400s with the invention of the printing press. The German goldsmith Johann Gutenberg's famous invention made it much easier to create copies of written work without having to write each one out painstakingly by hand. Furthermore, because each copy was identical, information could be reliably found on the same page of each copy. This led to the concept of the index, while also allowing people to confidently cite each other's work.

This new found ability to search written works using the index and link to other works using citations meant scientists were better able to communicate with each other. Connections were strengthened and a network of thought gradually developed among the scientific community. As a result, knowledge spread and grew. This

led to ideas that would eventually challenge the authority of the day—the Church.

In 1543, Nicolaus Copernicus published *'On the Revolution of Heavenly Spheres'*, in which he put forward the theory that the planets and the sun did not revolve around the Earth as previously thought, but that Earth and the other planets in fact revolved around the Sun. Although the book caused only mild controversy at the time, it managed to land Galileo Galilei in significant trouble over 60 years later.

An Italian physicist, mathematician, philosopher and astronomer, Galileo Galilei is today most famous for inventing the telescope. He saw that there were moons that orbited around Jupiter and used this finding to support Copernicus' earlier theory that Earth revolved around the Sun. In 1611 he travelled to Rome to let the influential Church philosophers and mathematicians see this discovery for themselves.

Rome, however, was not impressed, judging Galilei's opinions dangerous and bordering on heresy. He was ordered not to teach or support Copernican astronomy. Despite the order, he persisted and after writing a number of books on the subject he was ordered to stand trial before the Pope on suspicion of heresy. Galilei spent the last years of his life under house arrest.

Nevertheless, knowledge and reason gradually replaced religion and superstition as the dominant forces in Western society, though literacy was still rare and usually only present among the clergy and the very wealthy.

It was not until the industrial revolution of the 18th and 19th centuries that production costs became low enough to allow books to be mass produced. It was only then that the literary skills of reading and writing became accessible to the population at large. As literacy grew exponentially, so too did the number of books being written. Libraries appeared in many communities to make this new wealth of knowledge available to the public.

Unlike the centralised splendour of the Library of Alexandria, in the days following the industrial revolution the ultimate store of human knowledge was not a single repository of rare scrolls but a large number of individual repositories each containing copies of various works.

However, these individual repositories of information, though useful in their own right, were nonetheless isolated. They would each be far more useful if they could all be connected in some way, making the entire contents of every library combined available to each individual branch.

On the other hand, if every book ever printed was available through any library, there would need to be a enormous list somewhere of every title. It would be too big to comprehend, let alone read through. If something that big were to be useful in a practical sense, someone would have to engineer a way to search through the list.

Such a search engine was created by Melvil Dewey in 1876. The Dewey Decimal System took on the mammoth task of organising all the world's information by breaking it into bits. Dewey developed ten broad classifications to cover any possible subject that could be written about. Each of the ten classifications had ten sub-sections, which in turn had ten divisions, each level of classification becoming more specific than the last.

In this way, a book on international law would be first classified as 3 for Social Sciences, then 4 for Law, then 1 for International Law, meaning it would be classified altogether as 341. To be further specific within the field, a decimal point would be used, followed by as many subdivisions as necessary.

Once the Dewey Decimal number had been established, all the books in that section would be ordered alphabetically by author and given a unique code or 'call number' not shared by any other book in the entire catalogue. For instance, a book called *'International Law*

and Global Climate Change' by Robin Churchill published in 1991 has the call number *341.762 1991 INT*.

Organising the sum total of human knowledge is no easy feat. Although by no means perfect, the Dewey Decimal System did deliver a practical solution and thereby lived up to the promise of the connected, searchable library network.

Knowledge, arguably, is more than just what is contained in books however. It is also people's thoughts, actions, beliefs and conversations. It is what we are thinking, what we are doing and what we have done. Knowledge is *everything* we know as humans.

Could it ever be possible to store all this information all at once? If so, how could it ever be organised? How could anyone possibly make sense of it?

The solution may never be perfect, but perhaps it could at least be practical. If it were possible, surely this would be the greatest achievement humankind had yet accomplished? The ultimate store of human knowledge, beyond what those before could ever have imagined. What would it look like? Would it even be visible?

So begins the story of the Internet, or as it sometimes referred to—the cloud.

Good things come in small packages

How do computers talk to each other? That is the question we must explore to discover how the internet works. Some of the cultural, political and economic effects of computers and the internet are explored later in this book, but to understand the effects of technology it helps to first understand the technology itself. Like binary, the technology which underpins the internet is at the same time sweepingly simple yet breathtakingly complex.

Imagine a large, thriving city. The inner streets, usually crammed with bumper-to-bumper traffic, have been closed off for a six-vehicle government motorcade as it makes its way self-importantly from one side of town to another. Even though it takes the motorcade an hour to make its journey, no-one is allowed to use any part of the route the motorcade will take until the entire trip is complete. From the perspective of those in the motorcade, this guarantees a fast, reliable and uneventful trip. For everyone else it guarantees a royal inconvenience.

Now imagine that everybody has a motorcade. Every trip anybody makes means closed roads. Far from smooth, uneventful trips, paths would cross, arguments would ensue and nobody would get anywhere. The only way to make such a system work efficiently would be if everybody

had their own private road to each place they wanted to go. Yikes.

Believe it or not, this is how the telephone network operates. In the early days of telephones you needed to have a separate phone for each person you wanted to call. Telecommunications technology became more sophisticated, but the principle is still the same. When you make a call, you are reserving a physical channel between you and the other caller. No-one else can use any part of the channel for the entire duration of the call, even if you are both quiet and it is simply silence being sent down the line.

The advantage of using this technology, called 'circuit switching', is that reserving the line allows for the most direct and efficient route for data to follow. The disadvantage is the reserved line is wasted when it's not being used, which is frustrating when there are only a limited number of lines available at any one time.

In the 1960s, computer scientists working at the Advanced Research Projects Agency (ARPA), a division of the United States Department of Defense, wanted to build a large-scale decentralised network. They wanted a system where data could travel between any two points of the network, allowing the network as a whole to survive should any individual point break down. Such a network is

not unlike a city's traffic grid, in which a car can travel from any address in the city to any other address in the city provided it knows where to go and how to get there. If a road is closed or unsuitable to travel on, it's almost always possible to find another route. This flexibility is what computer scientists wanted, but the existing circuit switching technology was clearly impractical. A new technology had to be developed.

Imagine our busy city once again, but this time we won't be closing any streets. There is still an important message to deliver to the other side of town, but no motorcade. Instead, six vehicles each take a numbered piece of the message and are told to make their own way to the destination. Each driver decides their own route, some traveling through inner-city traffic while others take a longer but less busy route through the suburbs. The vehicles arrive in a different order to which they set off, but because each piece of the message they are carrying is numbered, the complete message can be reassembled correctly at the destination.

This is how computers send messages to each other—by breaking the message into packets, sending them separately and then reassembling them at the other end. This technology is called **packet switching**. It was used to create ARPAnet, the first operational packet-switching network, which eventually led to the interconnected

network of computers—the 'internet'—researchers at ARPA had envisioned. The way in which packets are sent, received and routed is known as **Internet Protocol** (**IP**).

In the physical world, every place that can send or receive a package in the mail has an address. An address is what allows the mail service to know where to deliver a package, or where to return it if necessary. For this reason, no two addresses are the same. On the internet, each computer, or more accurately, each point of the network at which data packets can be sent or received, also has a unique address. An **IP address** doesn't sound as interesting or memorable as a physical address, instead it is simply a sequence of numbers, for example 203.144.30.101. Nevertheless, an IP address serves its purpose, which is to identify your internet connection point from the millions of others around the world.

All this is simple enough, if a little mundane. What makes the technology that runs the internet so amazing is the way data travels from one IP address to another. Data on the internet doesn't travel in a straight line. Instead, it takes a look at where it is and where it has to go and calculates a series of hops, traveling through other points on the internet as it goes.

Consider how this works in the physical world. If I am in Brisbane, Australia and want to travel to Rubicon

Estate, a winery in the famous Napa Valley, California, I don't go out my front gate and look for signs to Napa. I have to plan a route. This is a complicated task and involves several steps. Once I have made all the arrangements, my route may go something like this:

'Get taxi from my house to Brisbane Airport. Fly to Sydney. Get connecting flight to San Francisco. Get shuttle bus from San Francisco airport to closest rental car depot. Drive rental car from San Francisco to Napa Valley and follow map directions to Rubicon Estate.'

This is roughly how data travels across the internet, only it happens *in a fraction of a second.* Perhaps, instead of traveling all that way to visit the winery in person, I simply download a photo of it from the internet instead. The route may be something like this:

'Fly wirelessly through the air from my laptop to the modem in the office. Travel from my modem to my internet service provider's headquarters in Sydney. Travel along high-speed cable under the Pacific Ocean to network hub in San Francisco. Travel to Rubicon Estate's web host provider in Palo Alto. Ask for Rubicon Estate. Go to fifth floor, find computer with image file on it. Retrieve file. Return.'

That's quite a trip. How long did it take? The amount of time it takes for data to complete a return journey such

as this is called a **ping**. What's my ping for this trip? 0.33 seconds. Sure beats jet lag.

Think about how the internet works for a while and it might blow your mind. Then again, the internet is just a bunch of computers that talk to each other incessantly. Just like the best gossip networks, what makes it so powerful is not the conversation, it's that everyone knows everyone else and news travels FAST![*]

Are you being served?

Many people regard the internet and the world wide web as being one and the same, though in fact this is not the case. The internet by definition is simply an interconnected network of computers which understand each other by communicating using accepted customs called Internet Protocol (IP). Having a globally connected system of computers enables new communication possibilities, including email, instant messaging, file

[*] It's worth mentioning that while hard drive storage is measured in **bytes**, web speeds are usually measured in **bits**. Hence, a 12Mbps (megabits per second) internet connection will transfer about 1.43MB (megabytes) of data per second. It's another one of those annoying inconsistencies that comes from the ramshackle way computers have evolved.

sharing, online gaming, video conferencing and browsing websites. Of these activities, only the latter directly involves the world wide web. The web is just one part of the internet, not another name for it.

The Web, as one might infer from its name, is about the strength inherent in links. Just as the printing press hundreds of years ago resulted in a way to link books together (the concept of citation), the internet has resulted in a way to link different pieces of knowledge together—the **hyperlink**.

A citation is when one document references another. If you follow the citation, go find the appropriate book and look up the right page, you will see the original reference. A hyperlink works the same way but with electronic documents. The difference is that if you follow the hyperlink you will see the originating reference *immediately*. In both cases, the ability to link different pieces of information has greatly strengthened society and the pursuit of knowledge. And in both cases, it started with the scientific community.

Tim Berners-Lee, an English computer scientist working at the European Organisation for Nuclear

Research (CERN)* in Switzerland, saw great potential in the idea of researchers being able to hyperlink to each other's work using the internet. He worked on integrating the prototype hypertext system he had already created with various existing internet protocols and in 1989 the early world wide web was born. It was later made available to the public in 1992.

It is worth reflecting on the fact that while the printing press was developed in the 15th century, it was not until the 18th century that books were mass-produced and not until the 19th century that literacy became widespread. In contrast, the microprocessor, the personal computer and the world wide web all arrived within the last 40 years. It is the web which is driving up the rate of computer literacy and it is only around two decades old. Perhaps this explains some of the chaos which surrounds the world of computers. We will pick apart this chaos later by looking at the companies, events and ideas which shape the technology world, but for now let's continue to explore the technology itself.

Documents and associated files intended for the web are stored on a particular type of computer whose only

---

* 'What's the deal with CERN?' you might ask. It's in French - *Organisation Européenne pour la Recherche Nucléaire*. Hang on, that's OERN. However, before 1954 it was called *Conseil Européen pour la Recherche Nucléaire* (European Council for Nuclear Research.) When the name changed the acronym didn't.

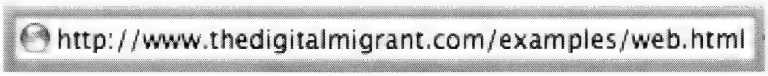
purpose is to take requests for information and send back the appropriate files. These computers are called **servers**. Offices, schools and some homes have servers too, but web servers are typically clustered together dozens or hundreds at a time inside purpose-built facilities with high-speed internet connections all around the world. If the internet can be said to exist anywhere physically, it must surely be inside these large data storage centres.

Racks of web servers

A computer which requests information from a server is called a **client**. In the case of a web document, the information exchange between client and server isn't handled directly but through a software application that acts as a **user agent**, usually a web browser. In other words, when you type an address into a web browser (user

agent), it makes a request on behalf of your computer (client) to fetch the information from the computer on which it is stored (server). The result of this request is then displayed in the web browser.

So far so good. But what does the gibberish you typed into the browser actually mean? We have already learned that every computer connected to the internet has its own unique IP address, but an IP address is just a bunch of numbers, not what you typed into the browser. Let's have a look at an example:



The *http://* part stands for Hyper Text Transfer Protocol, simply a grand way of saying 'hey, I want something off the web'. This tells the server that you're requesting a web document rather than, say, trying to send an email or start a chat with someone. The fact that you're typing it into a web browser makes your intention fairly obvious to begin with, which is why most of the time you can leave the *http://* bit off altogether and it won't matter.

Similarly obvious is the *www* part, which simply indicates what you're requesting is part of the world wide web. This distinction was important in the early days of

the web but has become less relevant since. Most servers will let you leave *www* off too.

The next part, *thedigitalmigrant.com*, is called the **domain**. This will help find the server's IP address. Because IP addresses are just numbers and difficult to remember as well as boring to look at, a system was developed to let people use names instead of numbers. Using this **Domain Name System** (**DNS**), a request for *thedigitalmigrant.com* will be directed to the nearest DNS server which will match it to the correct IP address.

Once the IP address has been matched and the server located, the last part, */examples/web.html* refers to the exact piece of information being requested from the server. In other words, telling the server 'I'm looking for the file called web.html. You'll find it in the examples folder.'

The entire string of information you typed into the browser has a name too. Whereas we might call it something straightforward like an address, computer boffins have decided to call it a **Uniform Resource Locator** (**URL**). With such a needlessly complicated acronym, it's at times like these that the scientific origins of computers are frustratingly apparent.

URLs are the links that make the web work. Some you may type in directly, but more often than not you'll

get to them by following links embedded in documents all over the web, like the ones that say 'click here'.

So that computers would be able to understand and interpret hyperlinks as well as other text formatting and layout within a document on the web, Tim Berners-Lee created a new language for the web called **Hyper Text Markup Language** (**HTML**). A document written in HTML is called a **webpage**, while a group of webpages is called a **website**.

As a computer user there is no need to be able to understand HTML—the web browser does this for you and displays the result the way it is designed to be seen. The reason for mentioning it is that while HTML is a fairly simple language, the way it is written by web programmers and understood by web browsers has profound and powerful effects on the way the web is used by ordinary people. One of the things it influences most is the idea of search.

How to win friends and influence people

When considering the internet as a resource, it's helpful to keep two words in mind—anything and everything. Any piece of information you can think of possibly exists and probably does. It's not just that anything can exist - *everything* can exist. There is no limit to the theoretical capacity of the internet. There are practical limitations in terms of the speed of the network, the number of available IP addresses and the capacity of data storage, but these can all be addressed by rapid advances in technology.

If the web is infinitely big, this raises an interesting question—how the heck do you find anything? If you know the address, no problem, but that only helps with information recovery—the things you already know exist. But if you have access to an infinite store of knowledge, what about *discovery*—finding things you don't know exist? When you have no idea what something looks like or where to look for it, how do you find it?

From the point of view of an end user, the answer is probably a search engine. But what about from the search engine's point of view? How is it possible to know about everything that exists? And even if you do know about it all, how do you lay your hands on it instantly? (I'm fairly

sure all the socks I own are in my room *somewhere* but that doesn't mean I know how to find them!) More importantly, how is a search engine supposed to know what you want if you're not sure what to ask for in the first place?

For libraries, this problem was solved by the Dewey Decimal System. If you want a book on Greek philosophy but you don't know what's available, you can go to the Greek philosophy section and have a look. You can even use the library catalogue to see what Greek philosophy books are available at other libraries. It doesn't always give you exactly what you want, but most of the time it gets you pretty close.

The catalogue idea doesn't work too well on the web however, as Jerry Yang and David Filo, the founders of Yahoo! found out when they tried it. In a library, when new books come in, someone has to add them to the catalogue and decide where they should be filed. That's exactly the approach Yang and Filo took in the early days of the web.

From their college dorm room, the boys kept an online list of their favourite websites, called *'Jerry and David's Guide to the World Wide Web.'* Their website became incredibly popular and it quickly grew to become Yahoo!—a full directory of the web that listed every

website by category. Although Yahoo! was by now a multi-million dollar company with oodles of staff, once the web went mainstream and usage exploded, Yahoo!'s directory couldn't keep up with the millions of new webpages being created every day. The web was growing faster than anyone could index it—simply too big and too fast for humans to cope.

In the late 1990s, not many people understood the importance of a good search engine—most in the internet business were concerned with getting rich and exploiting the new web craze. Yahoo! had become a portal—a destination website that aimed to provide visitors with everything they could ever need all in one place. Meanwhile, two different college students had seen the search problem coming. Coincidentally they were studying at the same university the Yahoo! boys had done just a few years previously.

Larry Page and Sergey Brin were excited by the web and its infinite potential but dismayed by how difficult it was to find anything. They realised the human approach wasn't working and although there were some automated search engines around, these weren't very good. The boys reasoned a technological platform like the web needed a technological solution to managing its information, something that could scale to match its size and speed. The system Page and Brin developed became instantly

popular. It was so good that in June 2000 Yahoo! even outsourced all its search facilities to the boys' new company—Google.*

There are two elements to understanding how Google's search engine works. The first is how it keeps an up-to-date index of everything on the web and the second is how it ranks what is relevant to you. The secret to both lies in hyperlinks—the way webpages link to each other.

Imagine you've just moved to a new school. It's your first day and you're uncomfortable and nervous because you don't know anyone and you can feel everyone staring at you. How do you get to know who everyone is? Assuming you think like a search engine and don't mind the fact you'd probably be beaten up, here's what you'd do:

Start a conversation with someone, maybe the person sitting next to you. Ask them to list the names of everyone they know at the school. Then go talk to everybody on the list individually and ask each one of them for the names of everyone they know. Then go talk to all of those people and do the same thing. Keep going until you run out of names. At this point you will be able to piece together a complete index of everybody in the school.

---

* A funky, deliberate misspelling of the word *googol*, meaning one with a hundred zeroes after it. The custom spelling meant the word was able to be trademarked.

This is how an automated search engine creates its index. Starting at any given webpage, a computer will scan the entire page and make a note of any links to other pages that it finds. It will then follow each links and scan those pages for more links and so on until it has scanned every page it can find. The result is a complete index of everything there is. Every page in the world is somehow linked to every other page—that's why it's called the web. If a webpage isn't linked with others in some way, it probably won't be part of the index. The computers that conduct this mammoth task are called **spiders** or **crawlers**. Google regularly conducts a complete crawl of the web, perhaps every few days, thereby picking up anything new or different since the last crawl. But Google didn't pioneer this technique, it's what they did next that made everybody sit up and take notice.

Back to school. You still don't know anyone—not surprising considering the only conversations you've had have been robotic, just a gathering of information with no emotional engagement. In fact, you're being stared at more than ever. But you do have a mine of information. While you were making lists of who knows who, you also took down a few notes about each person you spoke to. Nothing too personal, just a few key words such as which class they are in and what subjects they are studying.

Sifting through the information you gathered, a few interesting patterns start to emerge.

A lot of people seem to have Kim on their list. Looking at Kim's file you see you noted down that she is the school captain. Makes sense. Looking closer, it seems Kim is particularly well known amongst those people who you noted play sport. Turns out Kim is captain of the team that won the inter-school hockey tournament. Makes sense. Now you're intrigued about what else you might learn from your files. Since you have nothing to do for the afternoon, you decide to go through your lists of who knows who and make a tally—every time someone is mentioned they get a point.

Sure enough, Kim has the most points. She is obviously the most well-known person in the school. Taking a look at Kim's list, you are interested to see that most of the people she mentioned also have quite a lot of points. You get the impression that Kim is a fairly influential person in the school.

The person with the most points after Kim is Mel. You assume she is influential too but when you look at the people who mentioned Mel, not many of them are high on the list. Neither are the people Mel mentioned herself. It seems Mel is well-known, but not very influential. After looking through your notes, you discover Mel and five

other people were suspended from school a few months ago. Sure enough, the five who were suspended are both mentioned by Mel and mention her themselves. You assume the reason Mel is so well-known is that many people are aware of her suspension.

You decide to tweak your results a little so they rank better according to influence rather than simply awareness. Because Kim has the most points, you give everyone she mentions five points each instead of one. You also make a separate list of all the people studying each subject to find out who is most well-known in each subject group.

You discover Chris has the most points among people who study performing arts, so you give everyone he mentioned two points. Steve has the most points in maths so his contacts get two points each as well. After weighting the results in this way for every subject, the total points ranking looks very different. Mel, who used to be number two, now doesn't appear even in the top 30.

The next day at school, you show the points ranking to someone. They look surprised and tell you that you just named the ten most influential people in the school. They ask you some questions about specific people and events. After you answer them all correctly, the news spreads like wildfire—somehow the new kid knows everything about

us! Soon people are clamouring to ask you their own questions and find out their own ranking. They want you to compile rankings for all kinds of different combinations of things. Steve is particularly interested in your system. Kim invites you to sit with her and her friends. When you compile the index again a few weeks later, you're amused to find yourself at the top of the rankings. It seems you've now become the most well-known and influential person in the school!

Google's secret weapon is called PageRank after co-founder Larry Page who developed it. Google is forever tweaking it and understandably keeps the exact formula very much to itself, but the principle of how it works is similar to the school story. By counting the pages that link to your website, PageRank can estimate how popular you are without knowing anything about you. Taking the idea further, if well-known websites are linking to you then you must be very well-known and probably quite influential.

Being able to automatically rank relevance based on popularity was a big step forward in search technology, but it's not enough to guarantee the results you want when you search for something.

The holy grail of search is to be able to ask a question in plain English and get the answer you want, not

necessarily the answer you asked for. We want computers to understand what we *mean*, not just what we say.

But the language of computers is ones and zeroes, not words and grammar. Computer science continues to advance rapidly but we are still a long way off a computer that could pass for human—a computer that can truly pass the Turing Test.

What Do You Mean, What Do I Mean?

The introduction of the internet meant that for the first time on a large scale, computers could talk to each other.

But 'talk' is too generous a word. As far as the computers that make up the internet are concerned, 'talking' means zipping billions of chunks of information around at high speed. When you request a page on the web, your computer will dutifully go and 'ask' another computer for that information on your behalf. That computer will ask another which will ask another and so on until the information is retrieved and then passed back down the line.

Computers get your information for you, but they really have no idea what it is. Like parrots, they are merely mimicking speech with no understanding of its meaning.

It's worth thinking about just how difficult it is for a machine to make sense of natural human language. It's not as simple as learning what individual words mean or how sentences are structured in different languages. Learning the rules of language is one thing. Interpreting the intended meaning is something altogether different.

The rules, grammar and structure of sentences in a language is called its **syntax**. Every language has syntax, even computer languages like C++ and HTML. Using correct syntax ensures what you say will be understood.

If you're human, you can probably get away with not using correct syntax and people will still understand what you mean. Computers? Not so much. Like teenagers and bureaucrats, computers need to hear things *exactly* the right way before they'll do what you want.

Why can humans comprehend meaning while machines can't? There is an entire science dedicated to understanding meaning. It's called semantics.

Some people, especially myself, are regularly told off for quibbling over the details of what something means.

"It's just semantics," they say. I say how we interpret meaning is crucial. Here's why:

Little Sam was staying with his grandmother for a few days. He'd been playing outside for a while when he came into the house and asked his Grandma, "what is it called when people sleep on top of each other?"

She was a little taken aback, but decided to tell him the truth. "It's called sexual intercourse, darling." Then she explained all about the birds and the bees to him in detail.

Little Sam just said, "OK" and went back outside to play. A few minutes later he came back in and said angrily, "Grandma, it is not called sexual intercourse! It's called bunk beds!"

A computer would not understand this joke. It would not even understand what a joke is. How would a computer interpret the words 'on top of'?

I tried putting the entire sentence 'what is it called when people sleep on top of each other?' into Google. The top result was the website of a band called 'My People Sleeping'. The next few results were about sleeping disorders. There was a website about relationship counseling and, bizarrely, a page called '**Top** ways to avoid cat-related **sleep** deprivation'.

All of this in the Top 10 search results, as well as an advertisement for sleep medication. Clearly Google had no idea what I meant. This is because search engines largely ignore words like 'is', 'it', 'on', 'of' as they could mean anything. It depends on context, something of which computers have very little concept.

What search engines do understand are **keywords**, in this case 'people', 'sleep' and 'top'. Like a puppy eager to please, Google shoots off to trawl its entire index for anything that mentions any of these words. If it can find more than one on the same page, so much the better. It's considered a bonus if the words appear next to each other.

That's how we ended up with 'My People Sleeping' as the top result. It was the page out of everything in the index which best matched the keywords, and won the popularity contest described earlier, presumably by virtue of being a well-known band.

It's fascinating, but not very helpful. Many people have learnt to phrase their questions in a way that's more Google-friendly, similar to the way you might speak very loudly and slowly to a tourist asking for directions. In this way I might type my search as 'define: sleep together' in which case Google gives me an assortment of very precise explanations. Feel free to try it.

Because I know about computers, I know that I can type 'define: something' into Google to get an explanation of what it is. I also know I can use the '+' sign in front of a word to demand that Google find it specifically or a '-' sign to exclude words I don't want in my results. I know I can put phrases in quotation marks or restrict my search to just a particular website by prefacing it with 'site:http://myfavouritesite.com'.

There are many handy hints like these for using search engines, just as there are many handy hints for doing just about anything on a computer. But they're not written down anywhere, or if they are, you have to know where to look.

So how is it that I came to know about these things? How do you acquire that most magical skill—knowing where to look? The truth is I don't remember. And that is another important part of understanding semantics.

Psychologists talk about semantic memory, observing that in most cases, what we remember of something is only the gist of what happened or how we felt about it. We often confuse this with our episodic memory—the individual details of what happened.

Research into the meaning of memory has led psychologists to conclude, for example, that eyewitness accounts of crime scenes are unreliable.

One infamous incident which neatly demonstrates this involved Australian psychologist Donald Thomson, who as it happened was doing research on eyewitness testimony. Thomson was a guest as part of a live TV discussion on the subject, in which he described research showing that witnesses were influenced quite often by something as simple as the clothes the criminal was wearing. If someone else was wearing similar clothes to the offender they were very likely to be mistaken for the offender.

Later, Thomson was arrested on allegations of rape. He was placed in a lineup and identified by a victim as the man who had raped her. Thomson pointed out that the rape was alleged to have occurred at the time he was actually on live TV and so obviously he couldn't have done it.

Amazingly, despite this, the police pressed charges. They dismissed his alibi that he was in plain view of a TV audience and in the company of the other discussants, including an assistant commissioner of police. The policeman taking his statement sneered, "Yes, I suppose you've got Jesus Christ, and the Queen of England, too."

Eventually it was discovered that the real rapist had attacked the woman *while she was watching TV*. She had been watching the very program on which Thompson had

appeared and in her memory had mistaken the face of the rapist with Thomson's face from the TV, which she also recognised. Needless to say, Thomson was cleared of all charges.

This is a true story. The irony of it is staggering. Computers wouldn't appreciate the humour in it of course, but to be fair, in this case it's unlikely that Thomson did either.

We remember the gist of things, even if the details aren't always correct. The gist is usually all we need to convey what we mean. Computers don't understand what 'the gist' is and this is probably the biggest cause of frustration for people who aren't used to them. Computers are accurate and precise. Humans are vague and incomprehensible with all their gists and multiple layers of meaning. Who knows what they're really talking about half the time?

Nevertheless, we don't want to have to learn to talk like a computer. We're no longer satisfied with computers that are merely fast, accurate and precise. Now we want computers to learn to understand *us*. And not only do we expect computers to understand what we say, we expect them to understand what we really mean when we say it.

If a computer is ever built that can do this, women all over the world will no doubt force their husbands and boyfriends to buy one as a translation aid in relationships.

It may sound hopeless, but there are people working on it. Tim Berners-Lee, the creator of the web, lay down the gauntlet for a fresh generation of computer scientists in a speech he gave in 1999:

"I have a dream for the Web [in which computers] become capable of analyzing all the data on the Web – the content, links, and transactions between people and computers. A 'Semantic Web', which should make this possible, has yet to emerge, but when it does, the day-to-day mechanisms of trade, bureaucracy and our daily lives will be handled by machines talking to machines. The 'intelligent agents' people have touted for ages will finally materialize."

The semantic web involves separating semantics from syntax. At its most basic level, for instance, it means understanding that 'I love ice-cream' and 'I ♥ ice-cream' are two phrases that both mean the same thing, even though the syntax is different. It means coming to grips with context.

Google and other search engines understand the problem and are already working on individual ways to address it.

For example, consider a search for 'tiger'. Do you mean the animal tiger? Or do you mean Tiger Woods the golfer? Tiger Airways? Tiger Beer? Tiger the operating system made by Apple? If one word is all Google has to go on, it has no way of knowing what you want and the search results you receive will most likely reflect that.

However, if you provide a little context, the results are often much better. I typed 'tiger us open' into Google. Those three words individually mean very little. What's more, Google would usually ignore a word like 'us' altogether because it's too vague.

Despite this, Google knew I was referring to Tiger Woods the golfer. All 10 results on the front page were about Tiger Woods. Google even suggested some related searches I might try - 'tiger knee surgery', 'tiger san diego' and 'retief goosen tiger', among others. These are all very relevant to me. San Diego is where the US Open was held when Tiger Woods hurt his knee. Retief Goosen was Tiger's fellow competitor who accused him of faking it.

Google has been combing through the millions of searches it receives every day looking for patterns and meaning, trying to make sense of what people are asking and what it is they might really be looking for.

It's a small start, but it's something. What's interesting is that such attempts to clarify meaning on the

web have begun in much the same way that web search itself did—by hand. Google has staff whose job it is to clarify what things might mean, one word at a time.

As the founders of Yahoo! learned however, the scale of the internet makes doing things by hand obsolete. And in any case, we want computers to work out the meaning for themselves without us needing to spell it out to them.

Nevertheless, the internet, only a few decades old, has already become the greatest repository of all knowledge in human history. Like Deep Thought, the computer in *The Hitch-Hikers Guide to the Galaxy*, we may well have the answers, we just don't know what the question is yet.

To make sense of it all, we need to work towards a Semantic Web. Tim Berners-Lee says:

"If HTML and the Web made all the online documents look like one huge book, [the Semantic Web] will make all the data in the world look like one huge database."

A book is something you read. It imparts wisdom, but the process is one-way. A database is something you *use*. It is constantly being expanded, and with the right questions it can tell you almost anything.

In the end, it may be people who are the computers which power the ultimate database. Such a possibility is

suggested by an increasingly common technique called **crowdsourcing** which you will hear more about in Chapter 6.

But that's enough theory for now. It's time to take a look at how computers behave in the real world.

PART II:

The Real World

Chapter 4

Turf Wars

in which we explore the real differences between Windows, Mac and Linux operating systems and why compatibility problems exist

Beep Beep

The sign says 'OH NO!'. Who knows where it came from, but it sums up the situation fairly concisely. For a few seconds there is a forlorn glance for the benefit of anyone who might be around to appreciate the pathos, then begins the long fall to earth, a distant thud marking the end of yet another misadventure. For Coyote, this is just another day at the office.

Coyote and Road Runner certainly aren't friends, but neither are they enemies in the usual sense. They both want different things out of life, but even though there's a big wide desert out there, their paths seems to cross repeatedly. For Coyote, catching the Road Runner has

become a fanatical obsession, even though there are other things he could eat that would mean less danger to life and limb. Road Runner simply enjoys being the fastest, most elusive creature around and isn't above taunting Coyote in an effort to egg him on to further self-inflicted humiliation.

Interestingly, of the pair, it's Coyote that gets most of our sympathies. Perhaps this is because we know he will never really catch the Road Runner, so we are free to cheer him on. The Road Runner can obviously look after itself, and if it gets too cocky, well it probably deserves to get caught. Besides, the whole thing makes great theatre, so everybody wins. Especially ACME.

In this chapter we will explore the evolution of the computer industry and what it means for everyday users of technology.

Apple and Microsoft are two well known computer companies. They are certainly not friends, but neither are they enemies in the usual sense. Their paths cross often, but they do not directly compete. Apple makes its money selling hardware, Microsoft makes its money selling software. Both are egged on by technology journalists and zealous fanatics on both sides.

The whole thing makes great theatre, but how did it get to be like this? How will it end? Which one is the

winner? Which one is better? Does any of it actually matter? By the end of this chapter you'll be able to answer all these questions and more.

In Chapter 2, I described a computer's operating system as being like a host, perhaps at a party. What sort of party you end up having and whether or not you enjoy it depends a lot upon the host, so it's important to know what to expect.

When it comes to operating systems today, there are three main choices, each of which comes with wildly different personalities. The choices are Windows, Mac and Linux. Windows is made by Microsoft, the Mac OS (Operating System) is made by Apple and Linux is made by the geek community.

All three operating systems perform a similar range of tasks, each with its individual strengths and weaknesses. Most people tend to identify with one or another depending on their personality.

But the point of this chapter isn't to pick a winner or take sides, it's to explain the differences between each system so you will understand what makes each one tick and why there are significant consequences no matter which one you use. Let's begin with a bit of history.

Point and Click

Hardly anyone had a computer in 1984—now hardly anyone is without one. Windows, which did not exist in 1984, is now by far the dominant operating system which powers the world's computers. How did this happen?

Microsoft's big break came from a deal with IBM in 1981 to install Microsoft **DOS** (**D**isk **O**perating **S**ystem) on IBM's new range of personal computers. DOS was a command line operating system—there were no graphics, no windows, it was all about typing text onto a screen.

Apple's early computers also used a command line operating system. In the early 1980s, computers were selling well among hobbyists and electronics enthusiasts, but they didn't appeal much to non-technical people, who found the command-line interface intimidating.

Apple is widely credited with popularising the personal computer, which they did with a visual and user-friendly interface that 'mere mortals' could use. In 1984 Apple released the Macintosh with its new visual operating system—Mac OS version 1. Sales skyrocketed and Apple got rich. In doing so, they created a healthy market for software applications that would run on the Mac OS.

While Apple was building the original Macintosh, founder Steve Jobs made a visit to Microsoft founder Bill Gates in 1982 to demonstrate a secret prototype and encourage Microsoft to develop software which would take advantage of the Macintosh's new graphical user interface.

After seeing the prototype, Gates decided developing software for the graphical operating system was the way forward for Microsoft. The company developed a suite of productivity software for the Macintosh called Microsoft Office.

Microsoft was thrilled by the success of the Macintosh and the quality of the Mac OS. Gates famously said at the launch of the Macintosh, 'To create a new standard it takes something that's not just a little bit different, it takes something that's really new and really captures people's imagination. And the Macintosh, of all the machines I've ever seen, is the only one that meets that standard.'

Apple and Microsoft have always been two different types of companies with different motivations. In 1984, Apple made its money selling hardware. It didn't sell its software, it gave it away on every computer it sold. Apple's motivation was to invest in creating better software to sell more hardware. By comparison, Microsoft didn't make

hardware, it made its money selling software. Its motivation was to encourage its hardware partner at the time—Apple—to sell more computers so Microsoft could sell more software to run on them.

Gates' vision was a computer on every desk, each running Microsoft software. At first, Gates tried to convince Apple to license its Mac operating system to other computer manufacturers, but Apple wasn't interested. Apple's vision was making what the company called 'insanely great' products. The lure of the Mac OS was what enticed people to buy those products. If Apple let other companies sell the Mac OS, why would they buy Apple's computers over someone else's?

Today, the motivation of both companies remains the same. Apple makes great software to sell its hardware, while Microsoft encourages its partners to sell hardware so it can sell software to run on it. But a lot has changed in the meantime.

After being rebuffed by Apple, Gates decided Microsoft should create its own graphical operating system. Doing so would help loosen its dependence on the Macintosh platform while setting itself up as an effective competitor to Apple.

Microsoft's graphical operating system—Windows—debuted in 1985. It was a promising new development, but

unfortunately this first version of Windows simply wasn't very good. It felt rushed and unstable, with features that looked similar on the surface to the Macintosh, but didn't work as well in practice.

Microsoft persisted however, releasing Windows 2 in 1988 and then Windows 3 in 1990. It was with Windows 3 —six years after the launch of the original Macintosh—that Microsoft's answer to the Mac OS finally gathered some momentum.

Apple was not amused. It had originally licensed some of its Mac OS patents to Microsoft for use in Windows 1, but with each new release of Windows, Apple thought the features looked so similar to the Macintosh that eventually it launched a lawsuit against Microsoft for copyright infringement.

While the lawsuit dragged on, Microsoft secured an ever-increasing foothold on the market. It sold Windows licenses to a long list of computer manufacturers and soon Bill Gates' vision of a computer on every desk, complete with Microsoft software, was starting to look like a reality.

As for Apple, in the early 1990s, things weren't going so well. Steve Jobs had been ejected from the company in 1985 after a boardroom power struggle, and the company was being mismanaged. Apple had stopped pushing the envelope in technological innovation and was resting on

its laurels, while spending a great deal of resources fighting the legal battle with Microsoft over intellectual property.

Apple watched, bewildered, as Microsoft surged ahead in market dominance. Many thought Apple should have taken Gates' advice nearly a decade earlier to license the Mac OS.

By 1995, once-dominant Apple's market share had fallen to around 7 per cent. The company decided that it would license the Mac OS to other manufacturers after all, creating a market of Mac-compatible clones, but the decision turned out to be a disaster. Not only did the clones fail to have much effect on the flagging market share of the Mac OS, they began to cannabalise sales of Apple's own products as originally feared.

Worse still for Apple, it lost the lawsuit against Microsoft. The court found against Apple in all its claims, except, strangely, that Microsoft was infringing on Apple's copyright by using the same trash can icon. That's why today the Mac has a Trash Can and Windows has a Recycle Bin.

What about Jobs, Apple's enigmatic and charismatic former leader? Where was he while all this was happening? After being kicked out of Apple in 1985, Jobs had declared 'a pox on both your houses' and had started a

new company called NeXT, with the goal of making computers and software more advanced than anything on offer from either Microsoft or Apple.

But by the mid 1990s, NeXT was struggling. It had indeed managed to create the advanced next-generation operating system and incredibly powerful computers Jobs had envisioned, but no-one was buying them. NeXT's computers were too expensive and couldn't make a dent in the huge market share of Windows and the dwindling Mac OS.

Overtaken by Microsoft, outclassed by NeXT and being undermined by its own clone strategy, Apple realised its only hope of avoiding oblivion was to return to its roots and do something amazing and imaginative. But the company, managed for a decade by sales and marketing men in suits, had extinguished its creativity and was unable to pull the much needed rabbit out of its hat.

Apple decided to buy NeXT, with the intention of using the spiffy NeXT operating system to re-energise Apple's stale platform and drive a new range of products.

With the purchase of NeXT Apple also re-acquired Jobs, who, bursting with creativity and unrequited ambition, quickly cut a path through the beleaguered company's executive hierarchy and took over the reins in

1997. He immediately set about implementing radical changes in an effort to save the company.

One of the earliest and most controversial decisions Jobs made was to bury the hatchet with Microsoft. Many Apple fans and employees believed that Apple's dire predicament was Microsoft's fault, but at a public event in 1997, Jobs tried to diffuse this sentiment. 'We have to let go of the notion that for Apple to win, Microsoft has to lose,' he said.

The two companies cut a deal. Apple dropped the remaining legal proceedings against Microsoft and entered into a patent-sharing agreement which made it easier for the two to work together. Microsoft made a $150 million investment in Apple as a show of confidence and promised to develop its now ubiquitous Microsoft Office on the Mac for five years. Apple promised to install Microsoft's Internet Explorer web browser on all its new computers.

The two rivals had become friends, at least for the moment. To understand what happened next and the way things are today, we have to take a closer look at the actions of both companies and the effect they had on everyone else.

The Blame Game

There is a memorable scene in the timeless political comedy *Yes Minister* in which permanent secretary Sir Humphrey Appleby is facing a committee which has some difficult questions about wasteful government spending. Part of it goes like this:

> Sir Humphrey: It is not for me to comment on government policy, you must ask the Minister.
>
> Committee Representative: The Minister advises us to ask you.
>
> Sir Humphrey: And I am advising you to ask the Minister.
>
> Committee Representative: When does this end?
>
> Sir Humphrey: Soon as you like.

We can all relate to the truth of this. What it highlights is the issue of accountability. Most people would agree with the basic idea that each of us is responsible for our actions and should be held accountable for them. Governments and corporations are also supposed to be accountable for their actions but in reality it doesn't always seem that way. Sometimes we know who is responsible but there seems to be no way to hold them to account.

When it comes to operating systems, the two different approaches by Apple and Microsoft result in two very different experiences for the users of their products.

The seeds were sown in Microsoft's 1981 licensing deal with IBM. The deal contained a crucial oversight on IBM's part—it didn't allow for exclusivity. Microsoft gleefully exploited this opportunity as any ambitious young company would, selling MS-DOS licenses to dozens of other computer companies to install on their own products.

Suddenly there were lots of computers on the market being billed as 'IBM-compatible'. But there was a catch—the computers from each manufacturer were different, so Microsoft had to write a slightly different version of DOS for each one. The result was that 'IBM-compatible' could mean anything from 'works like a charm' to 'will read the disk but beyond that—good luck.' This was the beginning of a compatibility nightmare that many computer users have come to know and loathe.

On the other hand, by refusing to license its operating system to other manufacturers, Apple has always had the upper hand when it comes to quality control. Because Apple is responsible for making both the the operating system and the computer it runs on, it has a large degree of control over how the whole system integrates and is experienced by the user.

Microsoft sells its Windows operating system to a variety of hardware manufacturers, which in turn sell a wide variety of different systems using different components. Microsoft has no idea what kind of computer Windows will be running on—it could be any one of a million combinations.

Under this arrangement there will inevitably be glitches and incompatibilities that detract from a machine's usability, but who can you complain to? After a few generic troubleshooting steps, Microsoft will tell you to contact the hardware manufacturer. The hardware manufacturer will tell you it's a software problem and to talk to Microsoft or the shop you bought it from. The shop you bought it from will tell you the warranty doesn't cover software, but they'll fix the problem for a nominal fee. When does this end? Soon as you like.

Apple's machines aren't perfect, but they are easier to troubleshoot when things do go wrong and you know who to point the finger at if you need to complain.

Quality control isn't just about protecting profits, it's also about style. This is where almost all passionate argument about Mac and Windows stems from.

Apple is famous for its product designs and notorious for its attention to detail. Steve Jobs once accused Microsoft of having "absolutely no taste". But not

everyone values form as much as function. The most common complaint leveled against the Mac is that it's too expensive. Windows fans poke fun at the snobbery of Mac users and the 'Apple tax' they have to pay to be part of the Mac cult.

It's often said that a Mac is like the BMW of computers, where everything is finely tuned to provide a driving experience more than just a bunch of components assembled to do a job.

It's also been said that if Microsoft made cars, then occasionally, executing a manoeuvre such as a left turn would cause the car to shut down and refuse to restart, in which case you would have to reinstall the engine. Or that sometimes the car would die on the freeway for no reason. You would have to pull over to the side of the road, close all of the windows, shut off the car, restart it, and reopen the windows before you could continue. For some reason you would simply accept this.

In the end, style is of course a matter of personal taste, but Apple's obsession over quality control does give it another important advantage—the ability to adapt to change.

Some of the best features of an operating system rely on the co-operation of certain hardware. If Apple wants to

implement such a feature, it can easily roll out the accompanying hardware in its next model of computer.

If Microsoft wants to implement a similar feature, it first has to persuade dozens of its manufacturing partners to include the accompanying hardware. Some will, some won't. Of those who do, the hardware will vary, as will the quality of the feature. Even though ten people may all be using the same version of Windows, they may have ten different experiences. There's simply no way to guarantee consistency.

This goes some way to explaining why the features of the Mac OS are usually more advanced and better implemented than those on Windows.

Much of Windows is still built around code that's more than 20 years old, before the internet went mainstream. Poor software design has left Windows users vulnerable to a large array of security threats that have caused widespread corruption of data, loss of productivity and identity theft.

Microsoft has taken steps to resolve these problems, but most of its attempts involve band-aid solutions that come with financial and performance costs, rather than the fundamental design overhaul that its operating system needs.

Apple conducted this sort of fundamental design overhaul when it acquired NeXT. Instead of keeping its existing operating system foundation and adding new technology and features to it, Apple threw out its old system altogether and replaced it with a new, heavily modified version of NeXT's operating system, which it called Mac OS X.

Mac OS X debuted in 2001, and although it caused some inconvenience in the short term, it was a necessary move and one that has paid off handsomely for Apple and its customers.

Mac users now have a modern operating system built from the ground up to cope with advancements in technology over the last 20 years, especially digital media and the internet.

Mac OS X is a stable, secure and state-of-the-art operating system which Apple is able to build on rapidly and continually due to its quality control advantage.

Microsoft, by comparison, still struggles to deliver a similarly secure and capable version of its Windows operating system. Quite what this means for the future of Windows is unclear, but basic quality control flaws are certainly not a winning long-term strategy for any product, because no matter what the reason is for the

problems with its product, it is Microsoft that will eventually be held accountable for them.

iWay or the Highway

There are advantages and disadvantages to a tightly controlled system such as that which Apple has. Accountability is an advantage as we have already seen. So is security. But what about the right to dissent?

If you're a Mac user, you're used to things being done a certain way. Apple designs its computers and operating system to deliver an overall experience—a consistency that Mac users have come to expect, so much so that even third-party hardware and software developers design their products to match Apple's look and feel.

The appeal of this design nirvana is indisputable, but it means Mac users, software developers and hardware manufacturers are beholden to Apple's design decisions, and Apple has proven to be a fickle master. When Apple decides that white is the new black, everyone who makes a living in Mac-land has to come along for the ride or risk being the odd one out.

This presents a bigger problem than may appear at first glance. Design is more than just what something looks like, it's the way it behaves, how it's made and even *why* it's made. Apple holds all the cards when it comes to the design of its computers and operating system, and it usually holds those cards very close to its chest.

To the everyday Mac user, one direct frustration of Apple's tendency towards secrecy is knowing when to buy. The fear is Apple may unexpectedly announce a new model tomorrow which is better and cheaper than the one you paid for today. To software developers, Apple's shifting whims can mean much bigger frustrations, which in turn affect everyday users.

When Apple decided to switch from using one type of microprocessor (PowerPC) to another (Intel) in 2005, the change meant every single piece of software for the Mac platform had to be rewritten to work with the new Intel processors.

If software developers didn't rewrite their applications, those applications would not be guaranteed to work on the new Intel-based Macs Apple was selling. If you were a software developer, you couldn't afford to simply resist the change because pretty soon Apple would stop selling the 'old' PowerPC-based Macs and no-one

would be able to buy computers that would run your software.

Apple tried to make the transition almost invisible to users and as easy as possible for software developers, because naturally it was in its interest to do so. It created a piece of translation technology called Rosetta which allowed old PowerPC applications to run on new Intel-based machines. The translation meant software which ran through Rosetta would suffer from slower performance levels and a few bugs, but at least it would run.

Apple included the Rosetta feature in its Mac OS X operating system and turned it on by default to cause minimal disruption to users. Apple also provided engineering support to software developers who were making the switch, and facilities in Xcode—the software Apple provides developers to make their own software—to make rewriting their applications as simple as possible.

The transition went smoothly as far as most were concerned, but not for everyone. Apple's efforts to support developers during the transition were commendable and comprehensive, but they only applied to developers who had already made a similar transition just a couple of years previously, which in turn had followed another transition before that.

It has already been mentioned that Apple completely rewrote its operating system from the ground up to take advantage of more advanced technology. That transition from the 'Classic' Mac OS to Mac OS X also meant developers had to completely rewrite their applications, and Apple similarly provided support to developers and functionality in their products to make the ride as smooth as possible.

Once the transition from the Classic Mac OS to OS X was complete, Apple started adding amazing and groundbreaking features into its new operating system. It created Xcode so developers could start making use of these new features. Soon after, Apple told those developers they would need to stop using the software they were currently using to write applications. Instead, they would need to switch to using the newer Xcode, because that was where Apple was now focussed and future advancements would only be supported through Xcode. Sure enough, when it came to the switch to Intel, those who weren't on board with Xcode missed the boat.

All of these transitions were made with good reason, and the relentless pace of Apple's technological agenda means Mac users enjoy the most advanced consumer computing platform available. But as a result, developers never know what Apple is going to do next. Many developers live in constant fear of being trampled

underfoot, their applications replaced or made redundant by Apple's own technology. Every time there is a major change in direction, some developers can't afford to keep up while others simply can't be bothered. This applies especially to larger developers who write their software for multiple platforms and not just the Mac.

Adobe Systems, whose Creative Suite software includes Photoshop, Illustrator, InDesign, Dreamweaver and Flash, is a must-have 'killer' application for professionals working in graphic design, publishing, advertising and web. Apple's products have long been favoured among creative professionals and Apple continues to invest in adding features and functionality which appeal to creative industries. Adobe, however, is slow and sometimes resistant to taking advantage of those features.

The reason for this seems to be that Adobe decided long ago it didn't want to be held hostage to Apple's dramatic mood swings, no matter how brilliant its moments of creative genius may be. Adobe's Creative Suite product is huge and complicated and is used by millions of businesses, many of whom think Adobe's own changes are too fast and dramatic, let alone Apple's breakneck rate of advancement. Over the years, Adobe has been selling an increasing proportion of its software to Windows users and relying less on the Mac platform for

its sales. If catering to the less demanding Windows market means missing out on the latest and greatest from Apple, then as far as Adobe is concerned, so be it.

This is a problem for Apple. When it's Apple's way or the highway, some may choose the highway. If a developer like Adobe, with a killer application, decides to abandon the Mac platform altogether, Apple could lose a substantial portion of its user base, pushing the Mac platform down the slippery slope to obscurity and obsolescence that it so nearly fell victim to in the 1990s. What good is having the best playground if no-one comes to play in it?

The greatest threat to the Mac platform isn't security vulnerabilities or the risk of a competitor having more advanced technology, it's Apple itself. When Apple hits the right notes, the song sounds fantastic and everybody is happy. The company is certainly wiser and more careful after its near-death experience in the mid-1990s. But in any game where someone holds all the cards and bets the house on them time after time, well... house, cards, you get the idea.

Free Beer and Free Speech

Microsoft's struggle to match Apple's level of quality and innovation is not because it doesn't have the money or engineering talent, nor because its user base is too big. It's really a problem of motivation.

Apple is motivated to make exceptional software to sell its hardware, which is where it makes its money. Innovation and continual improvement are therefore part of the company's DNA.

Microsoft makes its money selling Windows licenses to computer manufacturers, not consumers. It has little incentive to make big changes to Windows because it knows the manufacturers will buy it anyway. After all, what alternatives do the manufacturers have?

Apple is not about to let anyone else sell its software —the secret ingredient that helps the company make a mint from selling the accompanying hardware. Licensing its operating system is what made Microsoft rich and powerful, but it proved disastrous for Apple, as the company discovered during its 'clone' misadventure in the mid-1990s.

One option available to computer manufacturers is to make their own operating system and sell a complete package like Apple does. However, it's not easy to compete

with Apple on quality and innovation nor with Microsoft on reach and distribution. As a result, would-be challengers, whether they be hardware manufacturers or software developers, are perhaps better off finding their own unique point of difference from which to launch an alternative.

One such alternative is Linux, which can boast something that neither Mac OS X nor Windows can—it's free. Linux is actually the name for a collection of operating systems which are all built upon the same basic code foundation—the Linux kernel.

Linux is free in more than one sense of the word. Not only is the operating system free of charge, it's also free to pull apart and mess with. Instead of an operating system which is developed, sold and supported by a large company, as both Mac OS X and Windows are, Linux is developed by a worldwide community of software engineers and released free to the public. Whereas a proprietary operating system keeps its inner workings (referred to as **source code**) secret, Linux lets anyone look inside and tinker with it if they wish.

Linux can be regarded as a sort of 'home brew' operating system. The Linux kernel is extremely stable and efficient, while just about anything can be built on top of it, making it incredibly customisable. This not only

makes it appealing to computer enthusiasts who like to tinker, but also to corporations that need a reliable, finely-tuned computing platform.

There are many flavours of Linux available, customised for different uses, called **Linux distributions**. The level of support available differs between distributions. While the Linux developer community is friendly, helpful and willing to provide support where possible, a corporation usually needs a level of support that is ongoing and reliable. Therefore, although each Linux distribution is free of charge, support often costs extra. Nevertheless, for many corporations, this is still an attractive option.

Microsoft appears to be terrified of Linux, and with good reason. Microsoft makes billions of dollars selling large numbers of Windows licenses to corporations and then selling support contracts on top. Linux threatens to undermine Windows by providing an equal or better product at a fraction of the cost. Microsoft has chiefly addressed this problem with a massive sales and marketing operation, including financing a range of independent studies in an attempt to show the 'total cost of ownership' of Windows is lower than Linux—an impressively creative if arguably disingenuous feat.

Why is Linux free? The short answer is it's about love rather than money. Linux is part of the Free and Open Source Software (FOSS) movement. As with any movement, the reasons and origins are complicated, but it boils down to the idea that a significant section of the technology community believes information and the tools to create it should be free and not 'owned' by anybody, instead belonging to the public. Perhaps this explains why Microsoft CEO Steve Ballmer has publicly referred to Linux users as "communists".

Other free and open source projects of note include the Firefox web browser, Wikipedia, OpenOffice (a free alternative to Microsoft Office) and WordPress (a free online publishing platform.) Microsoft, a company that makes money by selling software, is no doubt horrified by a movement that gives away its software free. Ironically, much of the energy driving the open source movement originates from people fed up with Microsoft's dominance in the technology world, which the FOSS movement sees to be stifling competition and innovation.

For example, the Firefox web browser was initially developed by a group of disgruntled former Netscape engineers. Microsoft was found guilty and convicted in 2000 by the US Department of Justice for using anticompetitive trade practices to drive Netscape out of business.

One of the major strengths of open source software is its freedom from ownership. Instead of confining development to a team within a single company and being subject to the constraints of a corporate agenda, open source projects allow the most capable and passionate people from around the world to develop software with the simple aim of making it better.

When it comes to open source software, Apple is not so worried. It makes its money primarily from hardware, not software, so it sees the free software movement as more of an opportunity than a threat. Like others in the tech industry, Apple often contributes staff and resources to various open source projects.

Corporate contributions greatly assist in the development of open source projects, while the efforts of many combine to produce a platform that the participating companies can use as the foundation for their own software. For example, Apple's Safari web browser and Google's Chrome browser are both developed from the same open source project—WebKit.

Microsoft's products appeal mostly to people who want cheap software that's good enough and can be used with a minimum of fuss. The problem for Microsoft is that this market is increasingly being catered to by open source alternatives. Linux is generally regarded as too fiddly and

technical for the average user, but there's a distribution gaining popularity called Ubuntu that's aimed squarely at the non-technical market. Some computer manufacturers are even starting to offer it on their systems instead of Windows.

Microsoft may not have the motivation to compete with Apple on quality and innovation, but it certainly has reason to be motivated by the threat to its business posed by the growing popularity of open source software. What the company does about it however remains to be seen.

Microsoft's biggest advantage at this point is its sheer ubiquity. But many of the people who use Microsoft's products every day aren't aware that *any* alternatives exist. If enough of these people discover that not only do alternatives exist, but also that many of them are better, cheaper or both, Microsoft is in big trouble.

Survival of the Fittest

Apple is the Road Runner. Unconcerned by the exploits of everyone else, it speeds along doing its own thing, content and perhaps a little smug in the knowledge that it can outrun its predators if need be.

Microsoft is the coyote. Cunning, determined and with seemingly unlimited resources, it often goes to elaborate lengths to catch up with the road runner, but as folklore would have it, never seems to succeed.

It's a game both have played for as long as they have existed, even though neither is forced to. At the end of the day it's a game that doesn't even matter very much. It's just a cartoon—predictable, though fun to watch.

For a long time, coyote and road runner have had the landscape to themselves, but the place is gradually filling up. The wild, untamed desert is becoming populated, colonised—domesticated. The biggest threat facing both coyote and road runner is loss of habitat rather than each other.

The explosion of the internet in the mid to late 1990s totally shook up the computer industry. The aftershocks are still being felt. Apple and Microsoft had both played a big part in kickstarting the computer revolution in the 1980s. They watched as a new generation of ambitious startups reshaped the technology world beneath them. They would have to adapt to survive.

Apple was mere weeks away from bankruptcy in 1997. Ironically, this gave it an advantage in such changing times—it had nothing to lose by taking big risks. As well as the deal it made with Microsoft, Apple

ruthlessly cut back its huge product range to just four distinct product lines based around desktop and laptop computers for home users and professionals.

Apple then took another big risk and released a bold new product—a brightly coloured all-in-one style computer specifically set up to get on the internet in minutes. The internet Mac—or iMac as it was called—was a huge success. An iBook laptop soon followed and before long brightly coloured plastic products were popping up everywhere. Apple had started a design trend.

At the turn of the 21st century, computers had become mainstream and the internet was opening up new possibilities for people to connect and share. Apple adopted a new strategy. It understood that as time went on, people would be using more and more technological devices. As people increasingly used the internet for their various activities, the operating system would become less central and exist more as a way for people to connect these activities.

Apple's new strategy was to be a 'digital hub'. Its new operating system, Mac OS X, was built with the internet and multimedia in mind. It created a suite of software called iLife which comprised powerful and well-integrated applications for getting creative with photos, music and movies. iLife was included with every new Mac. Then

came the phenomenon which launched Apple back into people's imaginations and gave the company a new lease of life—the iTunes/iPod combination.

iTunes started as a music jukebox application for making playlists and CD mixes—Apple ran an ad campaign at the time called 'Rip. Mix. Burn.'—but over the years it has become a well-organised repository for all sorts of media. The iPod was a portable version of iTunes that let people take their own personal repository of media with them wherever they went.

There were other music jukebox applications before iTunes and other portable media players before the iPod, but Apple took things a step further by launching itself into the media distribution business. It negotiated deals with record labels to sell music through the iTunes application, and later negotiated similar deals with TV networks and movie studios.

Apple has evolved from being just a computer company to being in the consumer electronics industry and the media industry as well. Nevertheless, its business model has stayed the same. Despite the huge amount of effort involved in running the iTunes Store, Apple makes very little money from it. But it helps the company sell millions of iPods, where it makes a killing. Once again, Apple is using subsidised software to sell its hardware.

As a bonus, people who had only heard of Apple because of the iPod started to become interested in the company's computers, spurring a new period of growth for the Mac. This became known as the 'iPod halo effect'.

The Mac was back, and Apple was firing on all cylinders. The roadrunner had almost been caught and eaten in the late 1990s, but since its near-death experience it has bounced back stronger and with renewed focus.

Microsoft, on the other hand, seems less sure of itself. By the end of the 1990s, the company had reached the peak of its success and influence. After the turn of the century, things started to unravel.

The Netscape antitrust trial was ugly and very public, and the conviction handed down in 2000 shattered the perception the company was invincible. The verdict signaled a change in public sentiment towards Microsoft.

The growing negative sentiment intensified as a result of the security problems that had begun to plague Windows users. The Windows operating system, built on code written a decade or more ago before the internet became popular, was not equipped to handle the viruses and other concerns the online world brought about. More on this in Chapter 5.

In an ironic twist, most of these security problems were a result of the close integration of Windows and Microsoft's Internet Explorer web browser, the very nature of which had helped the company drive Netscape out of business and for which it had been convicted.

Microsoft now also has to contend with Google, a young upstart born of the internet age which has already become the fastest growing company in history. Google, like many internet companies, has no love for Microsoft, and doesn't shy away from attempts to undermine the company's monopoly position.

Google makes almost all of its money from advertising revenue, which it uses to fund a dizzying array of free web services such as Gmail, Google Maps, Google Docs, Picasa and Google Desktop that further extend its reach. Microsoft struggles to compete with the Google onslaught, especially as Google—the new 'cool' place to work—continues to lure Microsoft engineers away to work at the 'Googleplex'. Such is the nature of Silicon Valley.

With the 2008 release of Google's own web browser—Chrome—many technology pundits have speculated Google's strategy is to take people's entire computer experience online and thereby make desktop operating systems irrelevant.

Though it has not said so publicly, the company appears to have Microsoft squarely in its sights. Google provides support financially and strategically to the Firefox browser which competes with Microsoft's Internet Explorer. Google supports Linux and uses it as the basis of its mobile phone platform. Google's CEO Eric Schmidt is on Apple's board of directors and the two companies regularly collaborate on projects. Nowhere does Google appear to support Microsoft. It develops applications for the Windows platform, but this appears mostly to be an attempt to white-ant the Windows operating system.

It's not all doom and gloom for Microsoft however. Although the company faces a daunting array of threats to its dominance, it has still managed to grow its profits each year. Like Apple, it too has diversified into new fields. Its Xbox game console is popular and enjoys considerable success through its efforts in game development. The company also continues to expand its reach and range of products in the corporate market.

However, Microsoft needs to redefine itself to ensure its long-term relevance. It is a huge and successful company with deep pockets and plenty of leverage, but it seems somewhat out of place in the internet age, lacking focus and drifting between new and interesting ventures and old, lingering problems.

Microsoft desperately needs to modernise its operating system with a fundamental rewrite, something it tried to do with Windows Vista but with disappointing results. The new Windows 7 is another opportunity to get it right, but in the meantime more and more people are looking at alternatives and they may be hard to bring back into the fold.

For computer users, things have never been better. There is plenty of choice in the market for computer systems and the internet continues to drive a golden age in technological innovation, ensuring it gets ever cheaper and easier to do more with technology that gets better all the time.

It's sometimes easy to forget just how far and how fast the computer revolution has come. Many of the headaches people get from using technology are due to compatibility issues between different systems and even sometimes different parts of the same system. While the technology boom continues, unfortunately these issues won't go away, but many things do get simpler and more reliable as various technologies mature.

Though we can't avoid having hiccups with computers altogether, at least an understanding of the differences between systems can help explain why hiccups happen, which in turn can help in dealing with them when

they do. It may also be interesting to note the word 'compatible' comes from the Latin *compati*, meaning 'suffer with'.

In the world of cartoon warfare, the characters will continue to fight each other endlessly, but arm yourself with a little knowledge and you don't have to get caught in the crossfire.

Chapter 5

Mostly Harmless

in which viruses and all manner of other computer security problems are explained

Pros and Cons

John took the last sip of his coffee and looked at his watch. It was time to go.

The little cafe he had discovered was perfect—the owner was friendly, the coffee was good and the food not too expensive. Since he first found the place a week ago, John had taken to stopping in each morning on his way into town.

"That'll be $13.90," said George, "Big day ahead, John?"

"Yeah definitely, it's the big meeting today," said John, patting his pockets. Damn! He'd left his wallet at home.

"George...," he said, "this is embarrassing, but I seem to be without my wallet. Is there any chance I can pay you tomorrow when I come in?"

"Sorry John, I'd love to, but my wife would kill me. Money's been tight lately and she counts every cent. I let someone pay me later the other week and they never came back. You can't be too careful these days."

John sighed and looked outside for a moment, thinking. He was going to be late. Suddenly he had an idea.

"What if I leave you my watch? I need to go home and get my wallet anyway, it's only down the road. I'll come back and pay you."

"Well sure," said George, "it's probably worth more than the meal anyway. I'll keep it here till you get back."

"I'll only be about 20 minutes," said John, handing it over. "See you soon."

John stepped outside and began walking home to fetch his wallet, walking quite fast because he was conscious of the time.

George put the watch on a shelf behind the counter and went to clear the table. A couple of minutes later the door jingled as a new customer came in. He was an older

gentleman and quite large, barely contained by his dark business suit. He ordered a coffee to go and then noticed the watch on the shelf.

"Is that your watch, sir?"

"Oh, that? No, it belongs to a customer. He'll be back soon."

"May I have a look?"

"I don't think so, wouldn't be right. Why do you want to look at it?"

"Because," said the old gentleman, "if it's what I think it is, it's very valuable." There was a pause. "I'm an antiques valuer," he added, by explanation.

"Really?" said George. "He will be pleased. You can talk to him yourself, he'll be here in a moment."

"Unfortunately I don't have the time," said the gentleman, taking his coffee, "I'm already late. But here's my card. Tell the owner to get in touch with me if he's interested."

"Just out of interest," said George, "How much do you think it's worth?"

"If it's genuine, around $50,000."

George was stunned. That was more than he made in a year! Suddenly he had an idea.

"I'm sure he wouldn't mind if I showed you the watch now," said George, taking it off the shelf. "That way at least he'd know whether it was valuable or not straight away. Here, take a look."

The valuer turned it over in his hands, taking an eyepiece out of his suit pocket and studying the tiny inscriptions and serial number closely. After a moment or two, he looked up with a smile.

"This is definitely the genuine article", he said. "Poor fool probably doesn't have the slightest clue what it's worth!"

"Poor fool indeed," said George thoughtfully.

"I really must fly," said the valuer, handing back the watch. "But give him my card. I'll be back through this way this evening."

George put the watch back on the shelf and watched the old gentleman leave. He stared at the card for a moment and then put it in a drawer. Not five minutes later, John came back, out of breath, opened his wallet and paid George the $13.90 for the meal.

George took the watch from the shelf. As he did so, he said "Nice watch, this. Do you mind if I ask where you got it?"

"My father gave it to me when he died," replied John. "It's not worth a lot, but it's worth something to me."

"Have you ever considered selling it?" asked George.

"Well no," said John, "why do you ask?" He was late and getting impatient.

"I have a brother who collects clocks and watches. This looks like exactly the sort he's missing and always talking about. Like you said, it's not worth much, but I'd gladly give you $1,000 for it if you were willing to sell."

A thousand dollars! John forgot about being late for a moment. Then he thought about his father.

"You know," he said after being silent for a moment, "they gave that watch to my father when he retired. He always wanted to buy a boat to sail in but he couldn't afford it—he was $5,000 short. I've never thought of selling the watch until just now when you asked. Never thought anyone would want it. But I reckon if I do sell it, it'd have to be for $5,000, for Dad's sake."

George made a show of being taken aback, but he kept in his mind what the valuer had said. $50,000! He'd

make a killing! His wife would get off his back for sure. Maybe they'd even go on that round-the-world cruise she was always harassing him about.

"You drive a hard bargain, John," he said. "But I'd never hear the end of it if my brother knew I'd seen the very thing he's been looking for all this time and let it go. $5,000 it is."

They agreed, and George had to go out the back and take the money out of the safe. There wasn't quite enough, so he made up the difference out of the week's takings. John looked at the money and the watch on the table and for a moment he seemed reluctant.

"I'll be sorry to see it go," he said. "Still, I think Dad would be proud of me."

"Of course he would," said George, now getting impatient himself. "Now here you go, you'd best get moving or you'll be late for your big meeting!"

Suddenly conscious of time again, John nodded at George, took the money and left without another word. George watched him go until he was out of sight. Then he stared at the watch for a moment and did a triumphant little dance. He got the old valuer's card out of the drawer, closed the shop and went to make a call.

A few blocks away at a different cafe, John sat down, took out the $5,000 he just got for selling the watch and slid it across the table to the large old man in the dark suit, who thumbed through it expertly. The old man looked at John and smiled.

"Well done, son," he said.

The Illusion of Security

Let's face it—the world isn't always a nice place or a safe place. Confidence tricks like the one you've just read have been around for hundreds of years—at least as long as there have been people around who are gullible enough to fall for them. But even being conned is preferable to many of life's other possible nasties.

You could be in a car crash, pursued by a stalker, be the victim of a terrorist attack or a random act of violence, be wrongfully arrested, be blackmailed, get cancer, fall off a cliff—the list is endless and terrifying.

We like to believe in the illusion of security, but the truth is there is always risk no matter where we are or what we do.

The celebrated deafblind author Helen Keller once said "Security is mostly a superstition. It does not exist in nature, nor do the children of men as a whole experience it. Avoiding danger is no safer in the long run than outright exposure. Life is either a daring adventure, or nothing."

I mention all this not to scare you, but because computer security is one of the most misunderstood issues in the world of technology and the one most likely to cause alarm.

Let's face it—the world of computers isn't always a nice place or a safe place. You can be conned, snooped on, infected and impersonated. You can be the victim of accidental or deliberate misfortune. Just like real life. There should be no illusions of security, but there are things you can do to prevent such misfortune, or at least significantly minimise the chances of it happening.

To minimise the risk of misfortune, instead of getting hysterical, get smart. Understand the different threats, where they come from and why, how they work and what you can do to avoid them. You'll feel smarter and safer as a result. If you let fear get the better of you, you'll miss out on some of the best things technology has to offer, and life in general for that matter.

Computer security doesn't just mean viruses—there's also phishing, firewalls, trojans, spyware, encryption, spam, identity theft and more besides. By the end of this chapter you'll know what all the important terms mean and what the difference is between them.

However, there is one word that sums up most computer nasties—**malware**—which literally means 'malicious software'. Malware is unique to computers and you'll hear about it in detail shortly, but first we'll take a look at some security threats that have their origins long before computers came on the scene.

Scams and confidence tricks have a long and colourful history. The techniques change and become more sophisticated over the years as various technologies evolve, but in essence they're the same tricks that have fooled people throughout the ages.

One well-known scam from the early 1900s is called 'The Spanish Prisoner'. In this trick the con-man tells his victim that he has been sent by a wealthy person of high social status who is in jail in Spain under a false name. This supposed prisoner can't reveal their identity without serious repercussions, so they are relying on the con-man to raise money to get them released. The con-man promises the victim he'll be rewarded handsomely if he supplies some of the money. Sometimes the victim is even

promised the wealthy prisoner's daughter to marry. Inevitably, once the victim and his money are parted, he learns that further difficulties have arisen which require more money. Such difficulties keep arising until the victim realises he has been conned, or presumably until he is totally out of money.

If that sounds totally implausible, consider that an almost identical scam has been doing the rounds via email for more than a decade and has tricked people into losing hundreds of millions of dollars. The United States Secret Service financial crimes division confirmed losses of over $100 million in one 18-month period alone. It's estimated that losses are probably even higher because many people are too embarrassed to report them.

I'm talking of course about the Nigerian Bank scam, in which the victim receives an email from someone who claims to know of a large amount of unclaimed money or gold which he cannot access directly. The details vary, but examples include the son of a deposed African leader who has amassed a stolen fortune, a bank employee who knows of a terminally ill wealthy person with no relatives, or a soldier who has stumbled upon a hidden cache of gold. The sums involved are usually millions of dollars and the victim/investor is promised a large cut of around 10-40% if they assist with retrieving the money, which is supposedly needed for fees, bribes, deposits and so on.

Naturally this money must be supplied in advance, and every time the victim thinks he is close to getting his cut, a new obstacle arises which inevitably requires more money.

Of the millions of people who receive such emails, only a very small handful actually fall for the scam, but those that do make the whole operation worthwhile for the scammers. The Nigerian Bank scam is one of the country's most lucrative exports, so much so that there are entire offices set up throughout the country for the purpose of creating and distributing false documents and impersonating the various people necessary to win the victim's confidence.

Although there have been international law enforcement crackdowns and awareness campaigns, the fact is that there are many scams out there and it doesn't take much imagination to think up new ones. As long as people fall for such tricks, con-men will keep exploiting them. Scams have been given a new lease of life thanks to the internet, because email doesn't cost anything to send and it's possible to reach millions of people worldwide. For the rest of us, that means more junk in the inbox.

Return to Sender

You may be constantly snowed under with junk mail, or you may get hardly any. Either way, you've probably heard the name that's been given to digital junk—**spam**. It got its name from the famous Monty Python spam sketch, which is hard to explain, but if you've seen it you'll understand. Spam is not just annoying, it's actually a huge global problem with significant social, economic and political ramifications, as you'll see.

Physical junk mail in the form of flyers, catalogues and coupons is wasteful and can be annoying, but some people enjoy looking through it and will occasionally follow it up. Despite the design, print and delivery costs, these follow-ups make it worthwhile for people to send junk mail.

Telemarketing works along the same lines, but most people find it far more annoying. Because telemarketing costs less than junk mail, it's possible to do more of it while at the same time needing less follow-ups to make it worthwhile for the telemarketers. The amount of successful responses per calls made is sometimes called the 'conversion rate'. Many telemarketers consider a conversion rate of around 1–2% to be worthwhile.

If you don't want to get junk mail, you can put up a 'No Junk Mail' sign. This is a social solution based on respect. There aren't really any social solutions to telemarketing. If you ask them not to call you there's no guarantee they won't call back another time or that you won't be called by someone else.

A technical solution would be to get a silent number that isn't listed in a public directory. However, this can cause problems among friends who rely on Caller ID to screen calls, because they will see your number as Blocked or Withheld when you call. Some countries—including Australia—have a legal solution which involves signing up to a 'Do Not Call' register. Unfortunately, not every telemarketer may be aware of such a register, or they may simply choose to ignore it and risk the consequences.

Email is the worst of all possible worlds when it comes to junk. There are essentially no costs involved, which means even a conversion rate of 1 in 10 million is worthwhile for a spammer. There are no social solutions. Asking a spammer nicely not to send you spam isn't going to work. In fact, it will probably mean you get sent more.

Spammers buy large lists of email addresses cheaply or use automated software to generate millions of random addresses, some of which will, statistically, be valid. The scale of waste involved is hard to comprehend, but it

means that if you reply to a spammer as an actual living, breathing human, you're doing the spam equivalent of standing on top of a large hill in a thunderstorm with a metal umbrella.

So just how big a problem is all this spam? It's estimated that spam accounts for 85–90% of all email traffic worldwide. All that waste puts pressure on internet providers to supply extra bandwidth, which raises the cost of internet access, while companies suffer from significant loss of productivity because their communication channels are so clogged up, which pushes their prices up too.

Many countries have enacted anti-spam laws that carry serious penalties, but, due to the global nature of the internet, legal solutions aren't particularly effective as spammers simply operate from areas outside those laws' jurisdiction. To compound the problem, spammers often use various types of malware to infect innocent people's computers so they give remote control to the spammer. These infected 'zombie' computers can then be used to send millions of messages on the spammer's behalf under a cloak of anonymity.

A common type of malware that gives spammers access to people's computers is called a **trojan**, named after the famous horse. A trojan usually takes the form of

an email or webpage which promises the reader something interesting such as a free screensaver, funny video or a document with hot investment tips. When the unsuspecting reader opens the file, it instead turns out to be a nasty piece of malware which changes the computer's settings to give remote access to the spammer.

A trojan may also contain a **worm** that then spreads itself to other computers automatically via the user's address book. Usually this all happens behind the scenes, so to the user it may appear as if nothing happened. A worm, once unleashed, can quickly enable spammers to create large zombie armies of infected computers (sometimes called **botnets**) by taking advantage of a spreading network of unsuspecting users.

Spam isn't just about email. It's also a problem on the world wide web. We've already discussed how search engines rank sites according to their popularity and number of incoming links. Spammers create phony accounts on various web services, often with the aim of influencing those search engine rankings. They set up networks of useless websites that are simply full of links to other sites, then post random automated comments on blogs, forums and message boards which contain links to these sites.

Their goal is to drive up their website's search ranking for popular search terms, even if the spammer's website has nothing to do with what someone is searching for. They then hope that a small percentage of people will visit their site and buy something, or perhaps fall victim to a scam.

The tactics employed by spammers are elaborate, wasteful and totally unethical, but ultimately cost the spammer next to nothing, meaning even a conversion rate of the tiniest fraction of a percent can constitute a profit. And when groups of spammers collaborate in organised networks, they can make serious money while causing serious disruption. As you can see, spam not only annoys people, it also causes financial harm and poses serious security risks.

Various solutions have been proposed, including charging everyone a fraction of a cent to send an email, which would make the economics of spam far less favourable, but would also mean a big inconvenience for everyone else. In practical terms, the most effective anti-spam solutions are technology-based.

Almost all email software now comes with a spam filter, while many internet providers also provide spam filtering services. A typical spam filter will first check the sender's address on an incoming email to see if it matches

anyone in your address book or with whom you may have had previous correspondence. This list of safe addresses is called a **whitelist**. If an incoming email is from someone on the whitelist, the filter will automatically let it through. Conversely, internet security firms and governments worldwide work together to keep up-to-date **blacklists** of known spam addresses, which are fed into spam filter software. If an incoming email is from someone on a blacklist, the filter will automatically block it.

If the sender isn't on either a whitelist or a blacklist, the filter will attempt to decide on-the-fly whether the email is spam or not. This dynamic filtering technology is based on complex algorithms and relies on the user to inform it if a legitimate message is blocked or a spam message is let through so it can improve its accuracy over time.

On the whole, email spam filtering has become quite advanced, and, save for the odd stray message, is an effective way for most people to manage the problem. It is however a game of cat and mouse, as spammers continually adapt their techniques to trick the filters and filters continually evolve to beat the spammers. A whole industry has arisen around the issue which keeps many people gainfully employed.

On the world wide web, while most online businesses have staff and systems employed to catch out spammers, perhaps the most common anti-spam technique is called a **captcha**. You may not know it by name, but you've probably seen one. Usually a captcha appears at the end of an online form—a little box containing funny looking letters and numbers that you're supposed to type into a box. This is to prove that you're a human and not an infected zombie computer spewing forth automated random gibberish.

A captcha works on the principle that if it generates a bunch of funny looking, barely readable letters and numbers, a human will be able to decipher it but a computer will just see a bunch of dots and won't recognise the shape of any of the characters.

In the time-honoured tradition of needlessly contrived tech industry acronyms, 'captcha' stands for **C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part. The Turing Test, you may remember from Chapter 2, is to tell if a computer can fake its behaviour convincingly enough to pass for a human.

Captcha technology is effective in stopping web spam, but it too is a game of cat and mouse. Spammers keep finding ways of beating the system, while captcha technology improves to outsmart them. Someone had a

stroke of genius however and realised that all this wasted effort could actually be put to good use.

In the quest to archive the world's information and make it accessible, there is an ongoing effort to digitise books, periodic journals and archives of old newspapers. This is mostly done automatically using scanners with Optical Character Recognition (OCR) technology, but it isn't 100% accurate. When the OCR software can't identify a word or gets it wrong, the scanned word gets put in a central database.

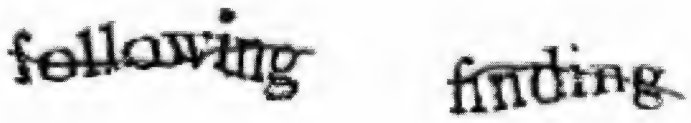
A new type of captcha was developed that uses the scanned words from this database to generate anti-spam images. Although OCR software can't accurately confirm some words, its best guess is recorded as the 'correct' answer.

As people identify these words and type them into captcha boxes all over the web, they may confirm the OCR software's best guess, or they may get it 'wrong'. If enough people get the same 'wrong' answer, that answer is eventually considered correct by the captcha software. If enough people enter a correct answer, the word is confirmed as correct and registered as such in the original digitisation project.

This is an example of the 'crowdsourcing' of information briefly referred to in Chapter 3 and which you

will see again in Chatper 6. While people are going about something ordinary and trivial—in this case using a captcha box—they are also assisting with the digitisation project. It's estimated that this process provides over 3,000 man hours of free labour every day. The technology was named **ReCaptcha**.

While spam and malware are still big problems, one has to bear in mind that the internet is still young, and the various good and bad phenomena that accompany it are fairly new to society. Nevertheless, elegant technological solutions such as ReCaptcha give a sense of optimism that we will eventually find similarly practical technological solutions to other security problems as well.



An example of ReCaptcha images, scanned from books in the public domain.

Safety in Numbers

When it comes to security, one of the most common questions asked is—how safe is it to use your credit card

online? The short answer is—safe enough. The long answer is, well, longer. But here goes.

For the purposes of this discussion it might be helpful to think of the internet as a giant postal service. There are messages zipping from one place to another all the time, each one of which has a clearly marked sender and destination. The postal service itself is public, but the messages are private.

One inherent security problem with the internet, as with the postal service, is that messages, while private, can be easily intercepted and opened. In theory, this makes the security of both systems incredibly weak. In practice this doesn't seem to matter much. We still send letters through the post and email through the internet, secure in the knowledge that with so much traffic hurtling by, our message isn't of enough interest or importance to anyone to make it worth intercepting.

On the other hand, not many people are comfortable with the idea of sending cash through the post. In theory, the most secure way to exchange money with someone would be to do it in person. However, this isn't always practical, so instead we might send a cheque or a money order. This adds an extra level of security by requiring a third party to certify that the money gets to its intended recipient.

In the case of a cheque, the third party is a bank. The sender authorises the exchange of funds with a signature on a cheque. To claim the money, the recipient must prove they are the person the cheque is made out to. Such proof usually takes the form of private identification such as an account number or driver's license.

This method is more secure than sending cash in the mail but still poses several risks. The sender's signature could be forged, as could the recipient's identity. Both sender and recipient must trust the third party with their personal details and their money. A bank employee might be lazy about checking signatures or identification, or a malicious employee might tamper with someone's details or money. The bank itself could fail to honour the transaction. Despite these risks however, in practice most people regard cheques as a reasonably secure method of transferring money, even though they may be inconvenient and somewhat costly.

Once upon a time, most people regarded cash as the most secure form of money. Over the years however, concerns about theft and personal safety compounded with the desire for greater convenience have seen people shift their trust to banks to look after their money for them. The banks are happy to oblige and even happier to charge a dazzling assortment of fees for the privilege of keeping people's money 'safe'.

As cash has become less common, money has become more theoretical. The most widely-used form of cashless money is the credit card, which is secured with the owner's signature. This poses the usual risks of forgery, theft or laziness in authentication, but is regarded by most people as secure enough in practice, not to mention incredibly convenient.

Considering just how fraught with peril traditionally accepted non-electronic transactions can be, the risks involved in using a credit card online are fairly trivial by comparison. The technology involved in processing online credit card payments greatly reduces the risk of forgery and entirely eliminates the issue of laziness in authentication. There is still the risk of your credit card information being stolen and used without your permission, but the important thing to know is that there is very little risk of this happening *online*. The biggest risk is someone stealing your credit card information in the offline world and using it to impersonate you on the internet.

The technology that powers the handling of sensitive information online is called **public key cryptography** and is beautifully clever in its simplicity. To explain how it works, let's return to our analogy of the postal service, where Person A (Alice) wants to send Person B (Bob) a secure message containing her credit card details.

Imagine Alice and Bob send secret notes to each other which they lock in a box with a combination padlock. For this to work, both Alice and Bob need to know the combination to open the lock. Unfortunately, there is no secure way for both Alice and Bob to know the combination without either arranging it in person beforehand or using a go-between, which means trusting someone else with the combination.

Public key cryptography neatly gets around this problem by not requiring both Alice and Bob to know the same combination. Rather than share a padlock they both know the combination to, instead they use two different padlocks.

If Alice wants to send Bob a message, she firsts asks him to send her his padlock, unlocked. Once Alice has Bob's padlock, she uses it to lock the box containing her message and then sends it back to him.

Bob is the only one who knows the combination to his padlock. When he receives the box, he opens it and gets the message. If Bob wants to send Alice a secret message in reply, he asks her to send him her unlocked padlock first, which he then uses to lock the box and send it back to her.

Using this method, neither Alice nor Bob need to know each other's combinations, and there is no risk of

anyone else intercepting the secret message because it is locked using a combination known to only one person.

It's possible for someone to intercept the unlocked padlock while it's in transit, but it's of no use to anyone by itself because it's impossible to deduce the combination by simply examining the lock. Both Alice and Bob are therefore free to send their unlocked padlocks publicly, each knowing they are the only ones who can unlock their padlock privately.

In terms of the internet, public key cryptography works in a similar way. When you use your credit card online, your web browser asks the merchant to send your computer a **public key** (the equivalent of the unlocked padlock). You then submit your details to the merchant in a secure package which is locked with this public key. The merchant uses its **private key** (the equivalent of the combination) to unlock your details and process the transaction.

So, what are the risks with this approach? It's possible someone could intercept the public key as its sent to your web browser, but it's no use to anyone without the private key which only the merchant has. Someone could intercept the message containing your credit card details, but again it's impossible to unlock it without the private key.

It's possible that someone could try to guess the private key by systematically using every mathematical combination possible, but the public/private key algorithms are calculated so that it would take around 1,000 years for someone to guess correctly using such brute force, making it a pointless exercise.

The biggest risks lie not with the technology but with other factors. It's almost impossible for someone to intercept your credit card details during a secure transaction, but if someone discovers your credit card details using some other means, there is nothing to stop them making purchases online. If someone breaks into the merchant's facilities and steals their customer database, your details could be compromised. These are risks that apply offline as well.

The only other significant risk is that you could be tricked into providing your details, either through a secure transaction with a shonky merchant or through a transaction with an impostor pretending to be a real merchant.

To combat the threat of shonky merchants, the concept of a **certificate authority** was developed. A certificate authority is an organisation that acts as a trusted go-between for customers to know they are

dealing with reputable merchants, and merchants to know they are dealing with legitimate customers.

A merchant can apply to a certificate authority for a security certificate that verifies they are a reputable merchant that will handle your details securely. The most widely-used certificate authority is an organisation called VeriSign.

All commonly used web browsers are programmed to check with VeriSign and other leading certificate authorities before commencing an online transaction, to see if a merchant passes this reputation test. If it does, a locked padlock symbol will appear in the browser to inform you it's safe to proceed. In most browsers you can click the padlock symbol to see details of the security certificate if you are still concerned.

That leaves us with the possibility of being tricked into a transaction with an impostor asking for your personal details. This is called **phishing** and unfortunately is a serious problem that costs individuals and businesses a lot of money. Phishing is difficult to control because it is a scam that relies on social engineering rather than technological security flaws.

Typically, a phishing attempt will come in the form of a spam email that looks like it could be from a reputable merchant or financial institution. The official-looking

email usually says something to the effect that you need to login to the organisation's website to 'verify your account' and provides a handy link you can use to do so.

If the unsuspecting victim clicks the link, he or she will be taken to a website that looks similar or identical to the organisation's official website, but which actually belongs to a scammer. Upon entering their personal information, the victim will usually be taken to the real organisation's website, but in the meantime their details have been sent to the scammer.

Thankfully, street smart users have little to fear from phishing. For a start, no merchant or financial institution will ever send you an email asking you to supply personal information to verify your account. Secondly, any correspondence you do receive from such organisations will almost always be addressed to you by name, rather than 'Dear Customer' or other such introductions. Thirdly, a phishing website is unlikely to pass the certificate authority test, so always check the padlock in your web browser before entering personal details. Finally, if in doubt, don't click the links. If you type the organisation's website address directly into the browser or use a bookmark in your web browser, you'll always be taken to the official site.

Many browsers have features designed to detect phishing websites and warn users about them, but it's best not to rely on technological solutions to what is really a social problem. Knowing what you now know about online security, you should be unlikely to fall for such scams.

Now that you've heard the long answer, you should also be able to decide for yourself how safe it is to use your credit card online.

Any Port in a Storm

Every system, digital or otherwise, has loopholes. And where there are loopholes, there will be people ready and willing to exploit them. In the world of computers, such people are called hackers. Not all hackers have malicious intent, the term simply refers to someone who likes to test the limits of systems and make them do things they weren't designed to do. Some hackers assist governments and security firms in finding weaknesses so they can make their systems more foolproof.

But for every well-intentioned hacker, there are others who are out to wreak havoc and exploit system vulnerabilities for fun and/or profit. Perhaps the most

well-known vehicle for such mischief is the computer **virus**. A virus is so named because it has two distinguishable qualities—it infects things and it spreads.

A virus starts life by being secretly planted inside a file, usually in a section of a disc or hard drive that is guaranteed to be accessed when it is inserted or started up. Once accessed, the virus automatically spreads to other parts of the system, infecting those files too. When those files are accessed, it spreads again. Viruses are usually designed in such a way that they quickly spread across networks to different computers, splitting and growing like bacteria.

Because a virus is usually hidden inside a host file, it can be hard to detect. Your computer may be infected with one or more viruses, and those viruses could be passed on to others without you even noticing. The same is true of **worms**, which are also capable of spreading by themselves, undetected. The difference between a virus and a worm is that a virus travels inside infected files, whereas a worm travels by itself, usually after being unleashed by opening an innocent looking file that is actually malicious (a trojan).

Many early viruses were in fact fairly harmless. Most were designed to inflate the ego of their creators by demonstrating their ability and bravado, in much the

same way that a graffiti tag artist enjoys infamy within a certain community. One of the very first computer viruses was called 'The Creeper', which displayed a message on infected computers that read 'I'M THE CREEPER: CATCH ME IF YOU CAN'. Other than display the message and spread itself, The Creeper virus did nothing harmful.

Today's viruses fall into two broad categories. Those in the first group are just out to cause trouble, usually by disrupting system files or destroying important data. Those in the second group are more insidious, altering settings and gathering information for the purposes of profiteering and organised crime.

As mentioned earlier in this chapter, one of the goals of viruses and other malware is to turn computers into zombies which can be used to send spam or further spread malware. Large numbers of such zombie bots are sometimes used to send millions of simultaneous requests to important websites such as government agencies or large corporations in an attempt to cause those sites to overload and shut down. This is called a **Denial of Service** (**DoS**) attack.

Of more concern to individuals is malware which secretly installs software that records whatever the user types into the keyboard and sends it to the malware creator. This kind of malware is called **spyware**. It is used

to steal credit card information, account passwords and other sensitive details as the user, unaware their keystrokes are being logged, types them in.

So how do you defend yourself against such nasties? It's an especially important question given that most of the time you're probably not even aware you're being attacked. There is no single, straightforward solution, but as was said at the beginning of the chapter, the best approach is to increase awareness while minimising risk. There are several ways to do this.

Firstly, keep your software up-to-date. All software makers regularly make updates to their products as they find and fix faults and security vulnerabilities. Often, by the time a new piece of malware makes it into the mainstream, the vulnerability it targets has already been detected and patched by the software maker. By keeping your software up-to-date you minimise the risk of being affected by these new attacks.

Sometimes however, a new piece of malware which the software maker is not aware of and has yet to patch will be make it into the mainstream. When this happens, the threat is called a **zero-day exploit**. These are usually the ones that make the news, as they can cause widespread infection and damage before they are able to be contained.

Another good idea to help reduce risk is to use software and tools that are less vulnerable to security threats in the first place. Most malware targets the Windows operating system. At time of writing, there is very little malware and no viruses that affect the Mac operating system. There is also little malware threat to other UNIX-based operating systems such as Linux.

It is widely assumed that the vast difference in security issues between Windows and Mac OS X is because there are more Windows-based computers than Mac-based computers in existence by a factor of around 20 to 1 and therefore Macs are not worth targeting. This is mostly an urban myth.

There are over 25 million Macs in use worldwide, particularly in significant areas such as education, scientific research and creative industries. One would imagine this would provide enough of a base to be considered worth targeting by *someone* at least.

A closer study of comparative security between operating systems leads to the more plausible conclusion that Mac OS X has inherent security advantages over Windows because of its more modern design and robust foundations. For non-technical users, Mac OS X is also a more secure choice than other non-Windows alternatives because it is owned and supported by a large commercial

entity—Apple—rather than developed and supported by the community as is the case with Linux.

It's important to remember that no system will ever be immune to security threats, so regardless of your choice of platform it would be wise to remain alert and exercise caution.

Of course, there is little point in having a great security system installed in your house if you leave all the doors and windows open when you go out. We've probably all heard the lecture about the importance of using different passwords for different logins and making sure those passwords are as complicated and unpronounceable as possible. No doubt we've all heard about the importance of keeping recent backups of everything too. Hardly anyone does either of these things, but everyone wishes they had once something goes wrong.

If your data or personal information becomes lost, damaged or stolen due to a malware attack, technological failure or even human error, it will be an incredibly frustrating and unpleasant experience. Don't wait until then to wish you'd been better organised. There are a variety of technological solutions for all systems which assist in making both backing up and password management a simple and efficient process.

After basic maintenance, common sense and good choices of systems, there are some technological options to help further minimise security risks. Of these, the most obvious is virus-scanning and malware-detecting software. An entire industry has grown up around and thrives on computer security issues, a fact that's worth bearing in mind when considering making an investment in such products.

Virus scanning and security software can be an effective way to help manage risk, though it usually comes with a price tag and some cost to convenience and system performance. However, if using Microsoft Windows, comprehensive security software should be considered essential.

Another technological tool to help manage risk is called a **firewall**. Your computer is like a transport hub for different sorts of digital traffic which pass through various **ports**. A firewall acts as traffic control and gatekeeper for all traffic going in and out of these ports through the internet or across a local network.

Firewalls can be set to block or allow different kinds of network traffic in certain times or situations, or ask permission before granting access in or out. When properly configured, a firewall can go a long way towards stopping invisible malware threats.

Firewall technology is built in to all mainstream operating systems, though someone with an appropriate measure of technical skill is usually required to set it up for maximum effect.

Every system has loopholes, and at the pace at which technology moves, there will always be new ones opening as others close. Different people have different reasons for using technology and different levels of risk in doing so. Nevertheless, a selective implementation of the tools and techniques discussed above should help reduce anyone's risk to an acceptable level.

Who Dares Wins

The security risks posed by computers and the online world are nothing new. They're really just the same risks that have always been around, adapted for an evolving society. In light of that fact, it seems silly and futile to shy away from enjoying the benefits of new technology simply because of risks that were also present in older technology.

As Helen Keller said, life is a daring adventure or nothing. Security is mostly an illusion and in the long run

avoiding danger is no safer than outright exposure anyway.

Such statements are not intended to scare people. Rather, they are meant to inspire and build confidence. Making the decision to embrace the world of technology is a brave step and a rewarding one. Digital technology offers incredible new experiences and is a daring adventure in its own right.

There are dangers of course, and those dangers are real. The point is to not let danger stop you from embracing the experience. There is just as much danger in not embracing things and you'll end up facing it without the thrill or wisdom of a great adventure behind you.

Surely the best approach is to acknowledge the risk in every endeavour, do what you can to guard against it, then go forward with confidence and don't look back. The greater risk is not what might happen if you take the leap, but what you might miss out on if you don't.

Chapter 6

Six Degrees

in which we discover how computers are changing our world and helping society reconnect

Look, Don't Touch

The drums were beating again. You could never tell when they'd start up, but when they did it was sudden, loud and terrifying. Reece squealed in fright. He needed a safe place to go to for reassurance. His eyes darted left and right wildly, looking for his mother, but she was nowhere to be found. She was never around.

The drums were loud and pounding in Reece's head and he began to panic. He needed comfort, needed someone to hold him, but there was no-one. Nothing except the cold metal contraption in the corner—the ghastly wire thing that looked like his mother but wasn't. It had a bottle in it that gave him food, but that was no

comfort now, not here alone with only the nasty noises all around.

What he needed now was warmth and love. He'd been alone all his short little life and he craved touch. There was none. Reece threw himself to the floor, clutching himself and rocking back and forth. He let out a blood-curdling scream and did not stop until the noises went away.

In the 1950s, American psychologist Harry Harlow conducted his now famous experiments with maternal depravation in infant monkeys. In the original experiment, baby rhesus monkeys were separated from their mothers hours after birth and raised by two surrogate mother machines, both equipped with bottles that dispensed milk. One 'mother' was made of bare wire mesh, the other was covered in soft terry towel cloth.

Given the choice, the baby monkeys always preferred the cloth mother and spent more time with it, clinging to it and snuggling. This might sound obvious now, but at the time it flew in the face of conventional wisdom on child rearing, which said that children should be given little affection and ought to be weaned off the attachment to their mothers as soon as possible.

The famous founder of psychotherapy, Sigmund Freud, had been largely responsible for this sort of

thinking. His theory of sexual sublimation and polymorphous perversion said that boys suckling from their mother's breasts are actually acting on subconscious sexual impulses. Freud theorised that children pass through five stages in their development in which they become fixated on different objects as a way of expressing sexual desire. The first stage was breast sucking, or the 'oral' stage as Freud called it.

If Freud's theories were accurate, the monkeys in Harlow's experiments should have shown no preference for the cloth mother over the wire mother, as satisfaction was supposedly derived solely from the act of sucking on the nipple and the sustenance it provided.

Harlow modified his experiment so that the monkeys were given no choice of mother. Some were raised by the cloth mother while the others were raised by the wire mother. In both groups, the baby monkeys' physical development was normal and identical. Psychologically, however, they were worlds apart.

The monkeys who had tactile contact with the soft, cloth mother developed normally. They had access to an important psychological resource—emotional attachment. The monkeys who did not have access to emotional attachment were permanently damaged, unable to socialise even in later life. These monkeys were observed

to stare blankly, pace in circles continuously and mutilate themselves.

Harlow's controversial research and other studies in attachment theory have revealed something quite significant—people *need* touch. Such is the impact on development and the effects of its absence that psychologists argue the need for touch is as important as the need for food, water and sleep.

Touch is the idea I'd like to use for this final chapter about the ways computers and the internet are changing our society. Technology is often discussed in terms of what we can do today that we couldn't do yesterday. In this light, many have declared the internet to be the most important invention in human history since the printing press. Perhaps it is, but forgive me if I don't re-tread this tired old road.

To me, a far more interesting story is how the internet is helping society rediscover how to touch. It's somewhat telling that back in the 1950s when Harlow was doing his experiments, visions of the future in popular culture almost always consisted of a world full of flying cars and robots. It can be argued that visions for the future reflect our concept of an ideal society, so back then one might conclude that society's collective unconscious

thought the ideal society was one that was efficient, practical and individual yet totally impersonal.

For decades as a society we have lamented the rise of materialism, loss of community spirit, growing anti-social behaviour and the idea that we hardly know our neighbours anymore. Computers and the internet are often attacked as just the latest nail in the coffin of a healthy, connected society. I believe the truth is exactly the opposite.

Over the years, television, radio and other sorts of broadcasting have helped turn our society into one that absorbs rather than participates. Well known internet lawyer Lawrence Lessig calls this phenomenon the 'read-only' society, and claims the 20th century is the only period in history in which such a phenomenon has occurred.

Necessity is the mother of invention, and as a society we have identified on some level a need for greater connection. As a result, we have invented the internet. Many people think of a computer as a box, and further suppose that spending more and more time and attention on a box must be antisocial and unhealthy. But what if we think of a computer not as a box but as a window? Better still, a *portal.* A way of reaching into worlds we've never seen before or could imagine existed. People need touch

and emotional attachment. That's the promise of the internet. It helps us discover each other and, ultimately, ourselves.

In the 21st century, the internet looks nothing like it did in the early days, but it's starting to look much more like the original vision of its creator Sir Tim Berners-Lee—a powerful force for individual creativity and social change. That change is unfolding around us and its effects are gathering momentum. I hope this chapter will help make some sense of it all.

Blogging is not a swear word

One of the most challenging aspects of teaching is trying to see things through the eyes of your student. Your goal is to help them learn things they didn't know and to help them understand things in new ways. This means you first have to imagine what it would be like not to know something, which can be quite difficult if you do, in fact, know.

Occasionally, just when you think you've explained something perfectly, a student can derail a train of thought with an innocent question that totally stops you in

your tracks and makes you suddenly realise you've been doing it all wrong.

Over the years, I have spent considerable time teaching small business owners to make better use of technology, helping them understand the role of technology in business and become more comfortable with using it. For several of these businesses, I have helped set up a **blog**, which in many cases is cheaper and more effective than the printed newsletter they'd been using.

Everything was going well in my world and I was feeling quite pleased with myself until one day I was asked an innocent question—"isn't 'blog' a *swear* word?"

I kept talking for a few seconds until my brain froze and I rewound what just happened. Did they just ask if blog is a swear word? Do I laugh? No, they're looking at me waiting for an answer. This is a serious question! What am I supposed to say? We've been talking about blogs for *two weeks*, why are they just mentioning this now? Have they learned anything? Do any of my students learn anything? *Am I a useless teacher?!*

Like a poorly programmed piece of software, my brain had been hit with something it wasn't expecting and shut down. My face was frozen in a goofy rictus of what must have appeared to be a disturbing mix of amusement

and horror as I realised my student was still waiting for me to answer the question.

Blog, of course, is not a swear word. It's simply short for **web log**, which is really just an online diary or journal. People have blogs to talk about their lives and their interests, and to share with whoever is interested the details of what they are doing or thinking about. Why did my student think blog was a swear word? I was too polite, or perhaps too embarrassed, to ask, but I spent some time afterwards wondering about it. If 'blog' actually was a swear word, how would it be used?

I'm just going to download the blogging software for you...

Ok, finally the blogging software is installed...

You know all that stuff you usually put in your newsletter? Blog it.

My client must've thought I swore like a sailor. I asked some of my other clients what they thought of the word blog and was surprised to discover many of them had similarly negative feelings about the word. How had I missed this?

It was a defining moment. I realised it's not enough to teach someone how something works. You have to give them a sense of context too. You have to teach them what

it means and where it comes from, why it *matters,* which means first you have to understand where they come from.

There's a recent TV show called *Mad Men* which is set in 1960 and revolves around the world of New York advertising executives. *Mad Men* is well regarded for its accuracy in depicting the period and one of the major themes of the show is to highlight the differences in social attitudes and behaviour then and now.

There's an episode in which Betty, the wife of the main character, has a nervous breakdown while driving and crashes the car. She feels she doesn't know her husband, who seems distant, and she worries about being abandoned. Her friend has been gossiping about the divorced woman who has just moved in down the street, and while Betty is driving she sees the woman struggling to lift a box and worries it might be her doing the same one day. This sets off a nervous condition where Betty's hands are cramped and, unable to hold the wheel, she crashes the car.

Betty's doctors recommend she see a psychiatrist. Her husband doesn't like this idea and thinks psychiatry is full of quacks. He says her doctors aren't trying hard enough. Betty wants to see a psychiatrist but is anxious

about it. "My mother always told me that it wasn't polite to talk about yourself," she says.

And there we have it. Blogging may not be a swear word, but it's easy to see how people might consider it rude. What is blogging, after all, but an endless stream of talking about yourself?

How do you explain that blogging is more than this? That it's changing the way we relate to ourselves as a society. How do you explain that blogging is *important*?

We've already covered the way computers think and the language they use, the differences between platforms and what happens when large numbers of computers are connected to form the internet. We've covered the different diseases computers can catch and how to guard against them. All we have left to cover is how technology changes the way society works and what those changes mean, yet this may well be the hardest thing of all to explain.

For the beginnings of an answer, let's wind the clock back to 1888 and a man named George Eastman, an amateur photographer from New York.

Eastman had been interested in photography for a decade or so, but was frustrated at how difficult it all was. Photography itself was only a decade or so older than

Eastman. It had already been made cheaper and easier after William Fox Talbot discovered a process for making negatives, but as these were made of glass and had to be kept wet, the process was too complicated and expensive for the population at large.

Eastman developed a type of paper film that was emulsion-coated. Unlike the glass negatives, Eastman's film negatives were flexible and held on a spindle, so a whole roll of film could be sent to a specialist developer. This made the process cheap enough for the average person to afford, but Eastman sealed the deal by making photography *simple* enough for the average person to use.

Eastman sold the film pre-loaded inside a special camera he had built—the Kodak. "You press the button and we do the rest" became the slogan. In a flash, people started taking up photography en masse.

Suddenly there were more photos being taken than ever before. A lot of them weren't very interesting of course, and many of them weren't very good, but that didn't matter, except perhaps to some of the professional photographers of the time who felt their profession had been devalued.

Even though amateur photography was mostly just people taking photos of themselves and each other, these photos had value and significance, not just to those in the

photos but to others as well. Photography became a way to document people's lives and their interests, and to share with whoever was interested the details of where they had been or what they had seen.

Photography wasn't rude or self-indulgent. It was exciting and emotional. It was important. It changed the world. And it began with technology.

"You press the button and we do the rest" could just as easily be the slogan for a whole host of technologies that have sprung forth since the invention of the internet. Blogging is just one of them. There are many others and we won't cover them all, but they do all have something in common—they put power in the hands of the people who use the technology rather than those who create it or control it. That's the change, and it's profound.

Mad Men is a show about advertising men who dream up new ways to tell people what they want. The characters' own lives are fractured and false, but they do their best to look like the happy American family, just like the ones you see in their advertisements.

As someone who grew up using computers and the internet, the world *Mad Men* depicts is utterly foreign to me. But to many people, a world in which they are empowered by technology in their daily lives is equally as foreign.

The challenge is not to explain the technology, it's to explain why it matters.

Read All About It

There's an ironic saying you hear every now and then —isn't it funny how there's always just enough news to fill the newspaper?

We know that what's in the newspaper is only a small sample of the total amount of news happening at any given time. And let's face it—news could be anything, it's just stuff that happens that's of interest to somebody. We could never possibly know everything that's happening, so we rely on a newspaper editor to decide what's important and present it in a nice little package we can have delivered each morning.

The job of an editor is to condense, correct or otherwise modify unfiltered information. A newspaper editor decides what goes in the paper and in what order, with the aim of best informing the reader of relevant issues. A video editor decides what shots to use in what order to best tell a story. It's an important job that requires good editorial judgment. If it's done right, the

audience won't notice. If it's done wrong, the results can be jarring and ineffective. But it does raise some questions.

What about bias? It's impossible for anyone to be truly objective. By letting one person or a small group of people decide what's important, we are beholden to their judgment, which may be coloured by prejudice, experience or ignorance.

What about individual taste? You can please some of the people all of the time, and all of the people some of the time, but you can't please all of the people all of the time.

To people who grew up before the internet, these are not huge issues for the most part. Most people have been conditioned to let others make editorial decisions for them. The news is what's in the newspaper or what's on TV. If you want to watch a certain program, you have to be in front of the TV to watch it at this time, on this channel.

Before the digital age, the owners of information decided what people could access, when it could be accessed and where people could access it.

In the space of just a decade or so, the information revolution has totally turned this state of affairs on its head. Now it is individuals who can decide what information they want, when they want it and where they

want it. In the digital age, we are all editors of information.

Information has become a commodity. It's still impossible to know everything that's happening, but the internet makes it possible to know just about *anything* that's happening. That's an important difference. An editor whose choice affects many can't please all of the people all of the time with their choice of content, but an individual can please themselves with whatever content they want anytime.

You can fill a newspaper or a broadcast schedule, but you can't fill the internet. As a result, niche content has become much more popular. In the past, if a topic interested a small number of people, it simply wouldn't make the cut. On the internet, it doesn't matter if only a handful of people are interested in something, you'll still probably find it.

While in the past we may have regularly read one or two newspapers, on the internet it's likely we regularly visit one or two dozen websites, maybe more.

An editor's job is to make sense of lots of unconnected information. If we are all editors in the digital age, we're going to need new tools to cope with so much raw information. Fortunately, technology has provided some.

Instead of going in search of information, you can make information come to you. This is achieved through a technology called **R**eally **S**imple **S**yndication (**RSS**). Most websites have an **RSS Feed** attached which you can latch onto using a **feed reader** application. The feed notifies you when content is published to the website so you can see what's new without having to actually go there.

The feed reader application, the RSS feeds you subscribe to and most of the content itself is free and available 24 hours a day, 7 days a week. Subscribe to the RSS feeds of your favourite few dozen websites and you'll have a steady stream of free news on tap, tailored to your personal taste and available to skim through whenever you want.

Radio and television, too, have been freed from the constraints of the broadcast schedule. For a long time it has been possible to record shows to watch or listen to later, but what you can record has always been dependent on what's on the schedule. In the digital age, you can program your own schedule.

Radio and television shows specifically created for the internet are called **podcasts**. Just as you can probably find a website dedicated to just about anything you can think of, it's more than likely you'll find a podcast devoted to your favourite subject.

Podcasts are based on the same technology as RSS. You subscribe to them using podcast software and when new episodes are published they are automatically downloaded for you and ready to watch or listen to whenever you're ready. And like RSS, the podcast software and the vast majority of podcasts themselves are free.

Traditional TV and radio programs and movies are also becoming easier to find online. Online stores like iTunes sell new releases and an extensive back-catalog, while many broadcasters have online viewing systems on their websites.

Of course, not everyone wants to spend all their time sitting in front of a computer, so *where* we can access information is just as important as what we can access and when.

If one product captures the spirit of the digital lifestyle more than any other, surely it's the iPod. A portable media player that lets you take music, photos, podcasts, contacts and calendars with you wherever you go, the iPod has managed to untether the digital lifestyle from the computer and take it to the streets.

Its creator, Apple, then took the obvious next step by creating the iPhone and added internet access, email and a phone, giving you all the power and purpose of a computer in your pocket.

It would be unfair to credit Apple with kickstarting the digital lifestyle, but as usual the company has identified a trend and created a great product which all but defines it.

That trend is mobility. It's not enough to have access to whatever we want whenever we want it. We demand to have it *wherever* we want it as well!

The change from just a few decades ago is staggering and is happening so fast that, like the rotation of the earth, you barely even notice it. But it is real change, and the impact on society will be widespread and noticeable.

Technology is empowering everyday citizens with the ability to access all the world's knowledge and manage it however we see fit. This is a first in human history and the significance of it can not be understated. It's a promise that began over 2,000 years ago with the Library of Alexandria and is just now unfolding.

In the digital age, we are all editors of information. We choose our level of engagement and actively decide what information we expose ourselves to rather than have it decided for us. And now that we are active in our engagement, many of us have decided that we want not only to consume, but to *create*.

Read/Write Culture

Creativity, in essence, is about constructing meaning. Everyone is creative to some degree if they have access to the right tools.

The Kodak camera put photography within the reach of the average person. At the time, people were amazed, but future generations took the technology of photography for granted. It became part of the language of everyday life. Digital technology is following the same path.

Pictures, sounds and video have become part of the digital domain. So has plain old-fashioned text. The digital world is nothing more than an endless universe of ones and zeroes. By making these forms of expression digital, they become theoretical. A picture isn't actually a picture in the way we usually think of it - it's millions of dots of colour represented by ones and zeroes. A song isn't a song —it's millions of ones and zeroes that make a sound wave. Video is the ones and zeroes of sound and pictures together.

A physically printed picture is made of dots too—dots of ink—but those dots are fixed and final. You can't change the look of a picture once it's printed. The dots of a digital picture however are flexible. The fearsome computing power that allows digital sounds and images to exist in the

first place also provides the means to manipulate them in an unprecedented number of ways.

Just as words and letters are the basic tool of creativity for the writer, pixels can be regarded as the basic tool of creativity for the digital artist. They can be twisted and shaped into any pattern, refined and remade in an infinite array of possibility.

In the digital world, text, sound, pictures and video are all the same thing—a multimedia language. The tools to create and publish using this language are tools that amaze those to whom they are new, but which generations of digital natives take for granted. Language has evolved and the idea of literacy is evolving with it.

Elizabeth Daley, Executive Director of the University of Southern California's Annenberg Centre for Communication and Dean of the USC School of Cinema-Television, says literacy is about empowering people to choose the appropriate language for what they need to create or express.

In his book '*Free Culture*', Lawrence Lessig describes a visit he had with Daley and Stephanie Barish, Director of the Institute for Multimedia Literacy at the Annenberg Centre.

Lessig writes of his conversation with Daley:

She describes one particularly poignant example of a project they ran in a poor inner-city high school in Los Angeles. In all the traditional measures of success, this school was a failure. But Daley and Barish ran a program that gave kids an opportunity to use film to express meaning about something the students know something about—gun violence.

"What you want is to give these students ways of constructing meaning. If all you give them is text, they're not going to do it. Because they can't. You know, you've got Johnny who can look at a video, he can play a video game, he can do graffiti all over your walls, he can take your car apart, and he can do all sorts of other things. He just can't read your text. So Johnny comes to school and you say, "Johnny, you're illiterate. Nothing you can do matters." Well, Johnny then has two choices: He can dismiss you or he [can] dismiss himself. If his ego is healthy at all, he's going to dismiss you. [But i]nstead, if you say, "Well, with all these things that you can do, let's talk about this issue. Play for me music that you think reflects that, or show me images that you think reflect that, or draw for me something that reflects that." Not by giving a kid a video camera and ... saying, "Let's go have fun with the video camera and make a little movie." But instead, really help you take these elements that you understand, that are your language, and construct meaning about the topic....

That empowers enormously. And then what happens, of course, is eventually, as it has happened in all these classes, they bump up against the fact, "I need to explain this and I really need to write something." And as one of

the teachers told Stephanie, they would rewrite a paragraph 5,6,7,8 times, till they got it right.

Because they needed to. There was a reason for doing it. They needed to say something, as opposed to just jumping through your hoops. They actually needed to use a language that they didn't speak very well. But they had come to understand that they had a lot of power with this language."

This is powerful stuff and the implications are profound. For just as it has never been easier to use the tools of multimedia to create, neither has it been easier to find an audience for that creativity.

The internet has sparked a renaissance in publishing. A new breed of multimedia-literate web services provides publishing tools that follow the Kodak model—press the button and we do the rest. YouTube, WordPress, Flickr—the names of these services vary but the concept is clear and consistent—free, direct publishing to the masses. It's simple and it's *searchable.*

Harnessing humanity's latent creativity and the internet's incredible search power, this cultural shift has thrown the established media models of supply and demand totally out of balance.

In the digital world, hordes of people are constantly creating anything and everything imaginable, meaning

there is an infinite supply of content. At the same time, people are finding anything they want using the internet's efficient search and information filtering technologies. Creators connect directly with consumers, and people who are consumers are increasingly becoming creators as well.

It's hard not to be excited by this giant leap in human potential, unless you're in a business which makes its living by connecting creators to consumers, in which case you are probably out of a job, or will need to drastically rethink the way you do things.

Music labels, movie studios, television stations, book publishers, magazines, newspapers—any industry whose business model is connecting content to audiences and raking a handsome profit off the top—all are struggling to compete with the internet, which does the same thing more efficiently and, largely, for free.

The internet and its associated technologies are still new, and the scope of the cultural change occurring is greater than most people realise. We are in a period of transition. But in the end, it's hard not to conclude that all these things—music, movies, television, books, information, news—will become applications of the web in some form.

Those who have seen the future are already embracing the internet. Some are resisting it. Others are

trying to exploit it. But regardless of whether people make use of the internet, there is a growing and distinct divide between those who understand the digital world and those who don't.

It's not enough to simply recreate old media culture using new technology. You have to understand the language of the new technology. Millions of digital natives do and their number grows every day.

As Elizabeth Daley says, "From my perspective, probably the most important digital divide is not access to a box. It's the ability to be empowered with the language that that box works in. Otherwise only a very few people can write with this language, and all the rest of us are reduced to being read-only."

Once again, it pays to think beyond the computer as a box and instead think about technology as a window, or a language. Either way, it's about connection with others. It's about touch.

What is touch in terms of technology? It's about understanding the ethic of the link.

The Ethic of the Link

It was fashionable at one point to refer to the internet as the 'information superhighway'. General tastelessness of the 1990s aside, a highway is quite a misleading metaphor for the internet. The internet doesn't go anywhere. It's not full of people travelling in the same direction. It doesn't stand out like a huge concrete construction that can be seen from space.

If you need a metaphor, then 'web' is as good as any. A web looks fragile but is surprisingly resilient. It's constantly being remade. It's full of beautiful patterns. Most importantly, every piece is connected to every other piece. In fact, what is a web except the sum of its links?

Links are what the internet is made of. Every new link makes the web stronger. You can't be part of the web without being linked to it. If you're not linked, you don't exist.

The same is true if you think of society as a web and people as links. A society is made of people. Every new person makes the society stronger. You can't be part of society without interacting with it. If you don't, you're an outcast and as far as society is concerned you don't exist.

In both cases, the more links you have, the better off you'll be.

In a society, everyone is connected to everyone else in one way or another. Hungarian author Frigyes Karinthy came up with the concept of 'six degrees of separation' in his 1929 short story, *Chains*. His musings fascinated future generations of mathematicians, physicists and sociologists, inspiring them to explore more of what would eventually be called 'network theory'.

What better network on which to apply the principles of network theory than the internet? It is, after all, the ultimate network, spanning societies across the globe.

Sure enough, with the right software, it's possible to create a map of everyone's relationship to everyone else.

What's this software called? You might know it as Facebook. Or MySpace. Flickr. Twitter. FriendFeed. LinkedIn. There are dozens of similar **social networking** applications, all with a slightly different focus and purpose, but all applying the same principles.

It's powerful stuff. Which brings us back to the significance of links.

A link, in one sense, is an endorsement. It says of the thing being linked to—this is important, or real, or true, or at the very least is worth being aware of. A link validates the existence of the thing being linked to.

Therefore, if you find something on the internet that is valuable or of use to you in some way, it's considered polite to link to it. Likewise, others will link to you if you are providing something of value to them. That is the ethic of the link.

Links are easy to make and cost nothing, so as a gesture, linking is not considered much to ask. The average blog post contains dozens of links to other websites. A social networking profile has links to the people who provide value in your life. Even an email forwarded 20 times contains the name and details of the person who provided that value to you before you sent it on.

Those who were in the business of providing access to information before the internet did not understand the ethic of the link at first. The newspaper companies and TV stations were accustomed to capturing an audience and selling it to advertisers. They were deeply suspicious of links, which take you away from where you are and send you somewhere else. "We don't want people to go somewhere else!" they thought. "We want them to come to us and stay with us!" This jealous guarding of a captive audience (or brand loyalty, depending on how you look at it) is the underlying principle on which TV networks have operated for decades.

This may be how things worked before the world went online, but on the internet this was a doomed strategy. The internet is a network that has the potential to contain anything and everything. The idea that any one source can provide all the value you will ever need is misguided at best.

Traffic on the internet moved onto other things. Blogs became popular as people started creating their own value and sharing it with others. New media outlets began to spring up that embraced the ethic of the link, and people started to turn to these sources instead, writing about them on their blogs and spreading the word around the internet. The more content that was created, the harder it became for the traditional media companies to keep people's attention.

People began to realise that to share is to gain. They discovered that using links to help people find value elsewhere earned them a reputation as someone to be trusted and valued in return.

Creating value for others helps nourish a culture where others are encouraged to do the same. As a result, everybody benefits. A rising tide lifts all boats.

If there is a cost to enjoying such a culture, it is transparency. It's very difficult to lie or keep secrets for long on the internet, because everything is linked to

everything else and almost all of it is public and searchable within seconds. Your presence on the internet results in a history of almost everything you do, easily researched and cross-checked by anyone with an interest. Try typing your name into Google and see what comes up.

Privacy advocates are rightly concerned about the long-term implications of such a world. Advocates for what is called **net neutrality** are keen to make sure that ownership and control of the internet remains in neutral hands.

To some, the reality of the internet conjures up images of an Orwellian nightmare. The well-worn cliche 'if you've done nothing wrong then you've nothing to fear' rings as hollow as ever. Call me an idealist, but I take an optimistic view.

We live in a society where we do not trust our leaders. We are deeply cynical about politics and politicians. We are media savvy, wise to the ways of advertisers, marketers and manipulators. We expect to be ripped off and we are always looking for the catch. We spend our lives protecting ourselves from those who want to take something from us, and we consider ourselves lucky if we escape with our dignity and our soul intact.

If ever there were a real Orwellian nightmare, we are already living it.

But imagine a culture where the main motivation is to create value rather than take it wherever you can. A culture where your reputation, easily fact-checked, is the most important currency you have.

Today we take a certain pleasure in finding out that people in power, whether they be politicians or celebrities, are involved in a public scandal. We indulge in such schadenfreude not because they did something wrong, but because they got caught. Serves them right, we think.

The truth is we all do things wrong all the time. We forgive others their transgressions because we know we are guilty of our own. In a culture of creation and transparency, such transgressions are more likely to be readily forgiven than pounced upon, especially if a would-be accuser's transgressions are equally transparent.

The more links you have, the stronger your place in the web becomes. The more we live publicly, the easier it is to see and analyse people's actions for what they are.

We come from a culture that encourages secrets, defensiveness and the illusion of security. Many of us are afraid of making new links and are fiercely protective of the ones we have.

The internet instead provides us an opportunity to embrace a different direction. There are no secrets in a

hyperconnected world, but we need to stop looking at that fact through the prism of the culture we are coming from. In a culture based on trust, we may just find that honesty, accountability and reciprocity follow.

If you're not convinced, I don't blame you. Neither am I, and I think to consider the internet a panacea for all society's ills would be more than a little naive. But I'd invite you to consider a couple of real-world experiments towards building such a trust-based culture. They are separate experiments, but inspired by each other, and it's hard not to find them encouraging.

Thought Experiments

Wiki is a Hawaiian word meaning 'fast'. Wikipedia, then, is perhaps best thought of as the fast encyclopaedia. This is because, unlike other encyclopaedias, anyone can contribute to it.

At time of writing, Wikipedia has 75,000 active contributors and over 150,000 volunteers in total, with over 10 million articles in 260 languages. It has a paid staff of only 23 employees.

"Imagine a world in which every single person on the planet is given free access to the sum of all human knowledge," says Jimmy Wales, Wikipedia's founder. It's a grand vision, and one which requires everyday people to do most of the heavy lifting.

You might think letting thousands of people write anything they want about any subject they can think of and calling it a reference source would be a recipe for anarchy, but this is far from the truth. As it turns out, given the chance to build their own ultimate reference source, the vast majority of people actually want it to be as reliable and accurate as possible.

One of the major advantages of having a reference resource built by the community is it draws on a wide range of knowledge and experience. Experts on any and every topic are encouraged to share what they know, and because there is a limitless amount of space, this sort of **crowdsourcing** approach makes Wikipedia a thriving compendium of incredibly comprehensive articles about everything from chemistry to cartoon characters.

With so much information on offer, consistency is obviously important. In recognition of this, there is a layout and structure common to all Wikipedia articles that makes browsing through articles an easy and comfortable task. Often, contributors will simply write what they know

about a given subject and it then falls to an employee or reputable volunteer editor to tidy up the article so that it conforms to Wikipedia's editorial standards.

Another key principle of Wikipedia is its interconnectedness. Articles are peppered with links to other articles, making it easy to glide from one topic to another without hitting a dead end or interrupting a train of thought. It's easy to get lost within Wikipedia for hours and emerge slightly dizzy with new information, but feeling enlightened nonetheless. There's also an extensive section of references and further reading links at the end of each article, making it easy to continue researching beyond Wikipedia itself.

Wikipedia's biggest boast though, or at least the feature that most consistently amazes people, is its speed. Whenever a major event unfolds, the fast encyclopaedia really comes into its own. Moments after a natural disaster, a terrorist attack or the death of a celebrity, the relevant Wikipedia articles will be updated, or a new one created which focusses specifically on the event in question.

As reports begin to flow in from eyewitnesses and media outlets, the page becomes a flurry of activity, with people constantly editing and re-editing information. It can be fascinating just to sit and refresh the page,

watching the shared consciousness shape and reshape itself.

Every edit made to a Wikipedia article is recorded, including details about who made the changes. It's possible to go back and look at any or all of the previous edits, right back to when the article was first created. This provides a practical level of accountability, and the community of volunteers is surprisingly diligent, even zealous, about fact-checking and correcting misinformation.

If the accuracy or reliability of an article is in dispute, as is often the case when it comes to events which are still unfolding, a warning message will be posted, essentially telling readers to take the information presented below with a grain of salt. If an article is repeatedly contested or vandalised, reputable editors can lock it altogether, preventing further edits until the issue can be resolved.

Although total accuracy can never be guaranteed, there are enough safeguards in place to make Wikipedia a reputable source of information, especially if it is used as the first point of reference on any given topic rather than the last word. In addition, Wikipedia's in-depth knowledge of obscure topics and the way it excels at keeping abreast of the times make it too useful to be ignored.

To me, the most astounding aspect of Wikipedia is its commitment to making knowledge free. You might wonder who owns the copyright on all the information contained in Wikipedia. Not the authors as it turns out, or even Wikipedia itself. Contributors agree to surrender the intellectual property rights to their submissions. Who is the beneficiary? The public.

All content on Wikipedia is released under a variant of something called the **General Public License** (**GPL**). In short the GPL means anybody is allowed to copy or reproduce all or part of the content however they see fit, even commercially, provided they obey a few simple conditions. The two most notable conditions are that the new content be attributed to the original author (the ethic of the link) and that it be released under the same General Public License.

The 'same license' condition ensures that ownership of the knowledge remains in the hands of the public. It's a clever use of copyright that essentially reverses what copyright is usually supposed to achieve, making sure content is continually in public hands rather than private ownership. Some people refer to this as **copyleft**. They're only half-joking.

The man responsible for copyleft and the GPL is Richard Stallman, a software developer born in New York

in 1953. Stallman grew up in the early days of computers, when software—both the application and the code used to create it—was free. Over time, software became commercial. Applications were sold for profit and the code used to create the applications (source code) was hidden to protect those profits.

Stallman was concerned by this. He believed that software was a form of knowledge, and that knowledge should be free, in the same way that mathematical and scientific theories are free.

In 1983 Stallman created the GNU Project, an operating system to be made entirely from free software. The GPL was written by Richard Stallman in 1989 for use with applications released as part of the GNU project.

Stallman's aim was to create a single license that upheld the principles of free software and could be used for any future project. Many of the applications created for the GNU project later made their way into the development of the Linux operating system.

Stallman also founded the Free Software Foundation and continues to advocate for free software today. As discussed in Chapter 4, the Free and Open Source Software movement is gaining in popularity and influence, even in mainstream use, as evidenced by the runaway success of the Firefox web browser.

The General Public License was designed for software projects, but the concept of copyleft has grown beyond that. Wikipedia, as already mentioned, employs the same principle for all its content. But there's an even greater experiment in free content underway.

Thanks to the explosion of creativity brought about by easy access to multimedia tools, there's more content being created than ever before. Most of it is amateur content. Not *amateurish*—some of it is of a very high standard indeed—but simply content created by everyday people for the love of doing it.

This sort of content is designed to be shared and remixed. If someone wants to reuse something you've made in their own creation, chances are you'll be flattered rather than angry, provided they give you a link.

The law states that whenever you create something, you're given copyright for that work. It doesn't matter whether you put the copyright symbol on it or not, the law grants you copyright over your work automatically. Whatever you create, you own.

Copyright law creates a culture of permission. You have to ask permission before you can use someone else's work as part of your own. Although it doesn't sound like much of a chore to ask permission before using someone's content, the physical process involved in *getting* legal

permission to reuse someone's content can be difficult, tedious and time consuming.

As a society we have a fairly entrenched view of creativity as property, so we might think this is no big deal. Consider this though—you don't need permission to dress up in a costume and sing a song you heard on the radio in front of your friends. But you do need permission if you make a video of yourself singing the same song and then upload it to YouTube.

It would be easy to spend time and money quibbling over the legal differences between the two, and indeed many have, voluntarily or otherwise. But the point here is that in both cases, singing the song harms nobody, yet the law punishes one and has no interest in the other. The law is a blunt instrument.

Many people would be happy for people to use and remix their amateur creations, but have no legal way to allow this without giving written permission individually to every person who asks. In the hyperconnected world of the internet, such a culture of permission stifles creativity.

As an antidote to this, the **Creative Commons** was born, founded by Lawrence Lessig in 2001. Under the Commons, people can elect to make their work available to others to use and remix without having to ask

permission first. They are, in effect, voluntarily surrendering some or all of their copyright.

The owner of the content uses one of several Creative Commons licenses to tell others how they would like to allow their work to be used and under what conditions. One of the main conditions is attribution, which ensures that whoever uses the owner's work will make it clear where they got it from and provide a link to the original. Other conditions can include non-commercial use only, or, like GNU and Wikipedia, a 'share alike' requirement that all subsequent uses are also released under the same license.

The goal of Creative Commons is to create a richer public domain. Instead of a permission culture, it seeks to foster a free culture. It's an experiment that is proving to be wildly successful, increasingly embraced even in the mainstream.

As just one example, in 2009, the Middle-Eastern TV network Al-Jazeera licensed its archive of news coverage from the Gaza Strip conflict under Creative Commons and made it available for anyone to share, subtitle, reuse, remix.

It's impossible to know where technology will lead us in the future as a society, but the trend towards a culture

where we connect freely and build upon each other's contributions certainly inspires hope and confidence.

In Western society, there's a lot of rhetoric shouted and lip service paid to the idea of democracy. The word democracy comes from the Greek word *dēmokratia*, where *dēmo* means 'the people' and *kratia* means 'power' or 'rule'. Democracy, then, is literally people power.

The internet is the greatest demonstration of people power in human history. It's still in its early days, only just out of nappies, but just like a child in fact, it has already brought us back in touch, not only with each other, but with ourselves.

Afterword

There is so much more that could be said, but that will have to do. I had originally planned to write 10 or 12 chapters on specific aspects of using computers, but the further I went, the more it seemed like a bad idea. If I'd tried to cover everything, there's no way I would have ever finished.

Instead I decided to tell a story. In doing so, the biggest challenge was not deciding what to say but deciding what to leave out. The result is not a comprehensive account of computers, but I've tried to capture the essence of what computers are about and I hope you put down this book feeling more confident and curious than when you picked it up.

Technology moves fast, as we all know. By avoiding too many specifics, I've tried to write the book in a way that won't date too soon, but undoubtedly things have

moved on since this was written and will continue to do so.

If I've done my job well, this won't be a problem. You'll know which questions to ask and where to go to find out what you need to know, or at least where to start. The most daunting mysteries of technology will be solved while the magic of possibility remains. Alternatively of course I might be tempted to write a revised edition if the need arises!

Being a migrant of any kind takes courage. It requires a certain strength of character to willingly enter the unknown and confront the consequences of such a move. Being a digital migrant is no different, and you should be applauded for making the effort to read this book. I hope it has been of some use to you.

The timing of our choices is as important as the choices themselves. At some point in the future this book will probably have to be renamed *The Digital Refugee.* By making the choice to go digital now rather than later, you are taking control of your future and that too should be applauded.

I probably haven't answered all of your questions about the digital world in these pages, and that is as it should be. I deliberately do not tell my students

everything at once. They learn far more by exploring on their own than they do during lesson time.

My favourite part of teaching is seeing what students accomplish off their own backs when given the basic tools of understanding and instilled with a sense of discovery.

You will enjoy exploring the digital world on your own. There will be times when it will all become too hard and confusing and you'll want to give up, but don't. Just remember what you've learnt, take a deep breath and think about what questions you need to ask to make sense of what's happening. Then go find out the answers.

That may sound simplistic, but it's really all there is. I've been using computers all my life but there are often still things I don't understand and find frustrating and impossible. I follow my own advice and usually find my way through it all eventually.

Occasionally even when I'm teaching, things will go wrong or something unexpected will happen. I used to worry this made me look stupid, but I soon discovered that people are mostly amused, even relieved, when something goes wrong for the instructor. It makes people feel better that it doesn't just happen to *them*.

I can't stress this enough. You are not stupid just because you don't know something. There's no way you

can know everything anyway. There's no need to be afraid or feel like a victim.

Unfortunately, people who know a lot about their chosen field sometimes become smug and disconnected from the very people who could benefit from their knowledge and expertise. Don't be bullied. Knowledge is power, so don't get angry, get even.

One of my students, while her son was away overseas, came to me after deciding she wanted to make a video presentation to surprise him for his 21st birthday. She had no experience with computers and was terrified of using them. Her son's birthday was in three months.

It was slow going, and everything had to be written down step by step, usually more than once. There were panicked phone calls and late night emails, and at one point she lost everything and had to start again. But she did it.

The result was a 20-minute movie, made of footage captured from old tapes, interspersed with the best of two decades worth of photos, tied together with titles and a musical score. Technology made it look nice and assembling it fairly simple, but computers had nothing to do with this mum's love for her son or her creativity in showing it.

Upon seeing the movie, her son was speechless. He couldn't believe that his mum, who in his experience had always been a total computer klutz, had made this all on her own. For that matter, neither could mum.

But she did. She has since set about touching up all the family's old movies and photos, bringing new life to old memories and having a blast in the process. Now she loves computers and is learning more all the time. She's even started showing her friends how to get creative with their computers.

Teaching is an often exhausting and frustrating profession, but it's stories like these that make it worthwhile. There's nothing quite like showing someone a new place and watching them get comfortable as they make it their own.

Thankyou for letting me be your guide to the digital world. Now it's yours to explore and enjoy. I can't wait to see what you do with it.

Good luck and have fun!

Acknowledgments

Many people have helped in the creation of this book, though I suspect only some of them are aware of the fact. I think it's only fitting that they get a mention here—I only hope I haven't left anyone out!

Thanks must first go to my grandparents, especially my Granddad Haydn, who I have already mentioned. I am their only grandchild and it has to be said that without their tendency to spoil me rotten at every opportunity there is a good chance I may never have developed such an abiding passion for technology. Let this be some proof perhaps that a little spoiling isn't always a bad thing!

Starting from the opposite end now, I must also extend my sincere thanks to Peter Green, Luke Martin and Jason Whittaker, who between them did the lion's share of proofreading and for whose keen eye and sensible judgment in curtailing many of my more bizarre tangents

I am extremely grateful. Extra thanks are due to Jason for insisting on writing the blurb for the back cover after several drafts of mine were apparently still lacking in pizzazz.

Thanks also to everyone else who helped with proofreading—Kerry Miller, Steve Chilcott, Grant Steele, Dorothy McCormack, Robert and Lili Friend, Anissa Loades, Hayley Neumann, Tanya Ormsby, Steve Androulakis, Bill Journee, Jonathan Poh and of course my family who were forced through various forms of guilt to read the same material over and over again. Like Blackadder's aardvark, I'm sure if they ever see an actual motherboard they'd like to step on its damn north bridge until it couldn't fetch an instruction if its life depended on it!

Thanks especially to Luke Martin, with whom I worked closely for many years on computer-related projects and who has been variously an inspiration, a guide, a companion, a worthy competitor and a right royal pain in the arse, as all the really best friends are. I hope this book will be a lasting testament to all the work we did together over the years.

I'd like to give a shout-out to some special people who have helped inspire me or given me a critical kick at the right moment—Adam Lindsay for being a fantastic boss

and kicking away the last barrier to me getting serious about starting my own business. David Homewood for courageously helping bring my wild-eyed vision into life and giving me a chance to prove myself. Mike Prior, Andre Vosloo and Stephen Holmes for their support and confidence in the setup and maintenance of that vision, and especially Andre for his generosity after it all went a bit pear-shaped. Andrew Rowland and Tom Matthew for their vital assistance in finding and helping me retain so many great students. Dorothy McCormack for taking me under her wing during a difficult time and being a much-needed mentor and dispenser of wisdom, encouragement and measured criticism. Dorothy, you are a beautiful, learned and valued friend.

I must also give a mention to Anthony Agius, whose enthusiasm and work ethic often rekindles the magic of technology for me and whose commitment to honesty in the face of great temptation I find humbling and admirable. Thanks also to Jonathan Haidt, who will probably never read this but whose work I find inspiring and challenging in equal proportions and who found time to reply to my email with a few pearls of personal wisdom.

Of course, this book really would not have been possible without the opportunity to tutor so many willing and wonderful students, whose experiences, insights and innocent questions have greatly informed my methods

and with several of whom I have become good friends. There are too many of you to list here and I know teachers aren't supposed to have favourites, but every good rule needs an exception so here goes—Jann Trout, Glenn Weiss, Sue Westlake, Robert and Lili Friend, Guy Norris, Marina Krasniqi, John and Chantelle Winship, Justino Zoppe, Eddie Leon, Barry Hilson, Ros Giles, Alyson Hourigan, Dort Burley and Christine McDougall—to name just a few of the favourites I'm not supposed to have. Special mention must also go to Gayle Hampson, who I hope will realise is the mum I mentioned in the Afterword, and to Helen and Russell Heinecke. Helen was my very first private student. She had been a secretary once upon a time but had never used a computer before and was fascinated by the idea that you could move blocks of text around the screen without having to retype them all again. It is that simple observation more than any other that I have constantly come back to in my quest to make computers more accessible and exciting.

I have saved mentioning my friends until towards the end, mostly because I know they will be among the few people persistent enough to make it this far. Michael Banasiak, Ricky Brown, Jason Whittaker, Steve Androulakis, Alf Santomingo, Hayley Neumann, Luke Martin, Ella Danes, Joel Maher—what can I say guys? Your unwavering support and encouragement never

ceases to amaze me. Thanks for putting up with me. Others too remote, too new or too numerous to mention here, thanks as well. You know who you are.

Thanks once again to my long-suffering parents, who have had to live with the reality of my exploits and who have endured many projects and experiments of varying success and longevity, odd spurts of enthusiasm and peculiar flights of fancy, all while offering only kind words and the very embodiment of support and encouragement. You have my deepest love, respect and admiration and I know I would never have made it this far without you. Don't ever change.

Finally, to my partner Doug, thankyou. You entered my life at a low ebb, and despite having little to show for it, have given me nothing but the most precious love, loyalty and trust. Your faith in me is humbling and will never be forgotten. You always have the right words at the right time and know how to bring a smile to my face when I need it most. I love you with all my heart.

Appendix

The Original ASCII Chart

ASCII Chart

Decimal	Binary	Value	Description
000	00000000	NUL	(Null character)
001	00000001	SOH	(Start of Header)
002	00000010	STX	(Start of Text)
003	00000011	ETX	(End of Text)
004	00000100	EOT	(End of Transmission)
005	00000101	ENQ	(Enquiry)
006	00000110	ACK	(Acknowledgment)
007	00000111	BEL	(Bell)
008	00001000	BS	(Backspace)
009	00001001	HT	(Horizontal Tab)
010	00001010	LF	(Line Feed)
011	00001011	VT	(Vertical Tab)
012	00001100	FF	(Form Feed)
013	00001101	CR	(Carriage Return)
014	00001110	SO	(Shift Out)
015	00001111	SI	(Shift In)

ASCII Chart

Decimal	Binary	Value	Description
016	00010000	DLE	(Data Link Escape)
017	00010001	DC1	(XON) (Device Control 1)
018	00010010	DC2	(Device Control 2)
019	00010011	DC3	(XOFF) Device Control 3)
020	00010100	DC4	(Device Control 4)
021	00010101	NAK	(Negative Acknowledgement)
022	00010110	SYN	(Synchronous Idle)
023	00010111	ETB	(End of Trans. Block)
024	00011000	CAN	(Cancel)
025	00011001	EM	(End of Medium)
026	00011010	SUB	(Substitute)
027	00011011	ESC	(Escape)
028	00011100	FS	(File Separator)
029	00011101	GS	(Group Separator)
030	00011110	RS	(Request to Send) (Record Separator)
031	00011111	US	(Unit Separator)
032	00100000	SP	(Space)
033	00100001	!	(exclamation mark)
034	00100010	"	(double quote)
035	00100011	#	(number sign)
036	00100100	$	(dollar sign)
037	00100101	%	(percent)
038	00100110	&	(ampersand)

ASCII Chart

Decimal	Binary	Value	Description
039	00100111	'	(single quote)
040	00101000	(	(left/opening parenthesis)
041	00101001	)	(right/closing parenthesis)
042	00101010	*	(asterisk)
043	00101011	+	(plus)
044	00101100	,	(comma)
045	00101101	-	(minus or dash)
046	00101110	.	(dot)
047	00101111	/	(forward slash)
048	00110000	0	
049	00110001	1	
050	00110010	2	
051	00110011	3	
052	00110100	4	
053	00110101	5	
054	00110110	6	
055	00110111	7	
056	00111000	8	
057	00111001	9	
058	00111010	:	(colon)
059	00111011	;	(semi-colon)
060	00111100	<	(less than)
061	00111101	=	(equal sign)
062	00111110	>	(greater than)
063	00111111	?	(question mark)

ASCII Chart

Decimal	Binary	Value	Description
064	01000000	@	(AT symbol)
065	01000001	A	
066	01000010	B	
067	01000011	C	
068	01000100	D	
069	01000101	E	
070	01000110	F	
071	01000111	G	
072	01001000	H	
073	01001001	I	
074	01001010	J	
075	01001011	K	
076	01001100	L	
077	01001101	M	
078	01001110	N	
079	01001111	O	
080	01010000	P	
081	01010001	Q	
082	01010010	R	
083	01010011	S	
084	01010100	T	
085	01010101	U	
086	01010110	V	
087	01010111	W	
088	01011000	X	
089	01011001	Y	

ASCII Chart

Decimal	Binary	Value	Description
090	01011010	Z	
091	01011011	[	(left/opening bracket)
092	01011100	\	(back slash)
093	01011101	]	(right/closing bracket)
094	01011110	^	(caret/circumflex)
095	01011111	_	(underscore)
096	01100000	`	
097	01100001	a	
098	91199919	b	
099	01100011	c	
100	01100100	d	
101	01100101	e	
102	01100110	f	
103	01100111	g	
104	01101000	h	
105	01101001	i	
106	01101010	j	
107	01101011	k	
108	01101100	l	
109	01101101	m	
110	01101110	n	
111	01101111	o	
112	01110000	p	
113	01110001	q	
113	01110010	r	

ASCII Chart

Decimal	Binary	Value	Description
115	01110011	s	
116	01110100	t	
117	01110101	u	
118	01110110	v	
119	01110111	w	
120	01111000	x	
121	01111001	y	
122	01111010	z	
123	01111011	{	(left/opening brace)
124	01111100	|	(vertical bar)
125	01111101	}	(right/closing brace)
126	01111110	~	(tilde)
127	01111111	DEL	(delete)

About the Author

Danu is in his mid-twenties and currently lives in Melbourne, Australia. He has been helping people make sense of technology for years—ever since high school when he designed a computer course for his teachers.

He has designed and presented computer courses for schools, seniors and small business and has worked for several years as a private technology tutor.

Danu is passionate about empowering people to change their lives using technology. Throughout his work with computers, he has found that good, simple explanations and friendly guidance from a supportive tutor can make converts out of even the most entrenched technophobes.

This is Danu's first book, but he has enjoyed it so much he is sure there are more to come!

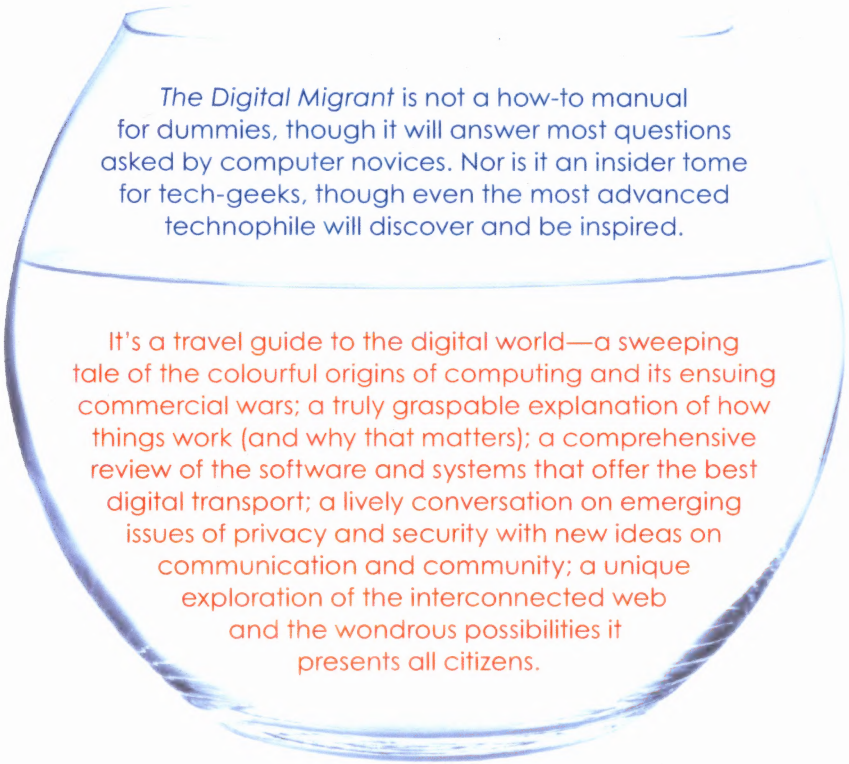
Still want more?

The journey continues at

www.thedigitalmigrant.com

THE FIRST STEP IN YOUR DIGITAL JOURNEY...

The digital age offers a wealth of opportunities for those willing to make the trip. Digital migrants are richer for the experience—in knowledge, access, and opportunity. Those who stay behind will become increasingly disconnected from the new world.


The Digital Migrant is not a how-to manual for dummies, though it will answer most questions asked by computer novices. Nor is it an insider tome for tech-geeks, though even the most advanced technophile will discover and be inspired.
It's a travel guide to the digital world—a sweeping tale of the colourful origins of computing and its ensuing commercial wars; a truly graspable explanation of how things work (and why that matters); a comprehensive review of the software and systems that offer the best digital transport; a lively conversation on emerging issues of privacy and security with new ideas on communication and community; a unique exploration of the interconnected web and the wondrous possibilities it presents all citizens.

Free of patronisation and prejudice, cutting through the techno-babble, *The Digital Migrant* is everything you need to know to participate in the technology age (without the stuff you don't).

Your digital journey begins with just one step—read The Digital Migrant.

ISBN 978-1-4092-8861-9

9 781409 288619

USA **$21.95 RRP**
AUS **$29.95 RRP**

Lulu **www.thedigitalmigrant.com**